

OPTIMAL MULTICAST TREES  
TO PROVIDE MESSAGE ORDERING

BY  
JAVIER E CORDOVA

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL  
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1993

## ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my advisor, Professor Yann-Hang Lee, for all his support and guidance throughout my graduate studies. Without his guidance and encouragement this work would not have been possible.

I also thank my other committee members, Professor Ravi Varadarajan, Professor Randy Chow, Professor Manuel Bermúdez, and Professor Chung-Yee Lee, for their interest in serving on my committee and for their valuable comments. I also thank Professor Tim Davis for willingly substituting for Professor Chow in the dissertation defense.

I would also like to thank the people from the University of Puerto Rico who helped me in pursuing my doctoral studies, and who gave me financial support through all these years. Thanks also go to the new friends I have met in Gainesville. Thanks go to Pedro, Sandra, Randall, Roger, Olga, JuanJo, Mafe, Luis, Jackqueline, Ronaldo, Fernando, Sonia, and many others who have helped to make this process much easier.

Last, but not least, I wish to express thanks to my family: to my wife Anneliesse, who supported me in the difficult decision of coming to a foreign country for so many years; to my children Natalia, Mariheida and Javier Andrés, who have given a new dimension to my life and a whole new reason to enjoy it; and to my parents Félix and María Isabel, who have supported me throughout all my career. To all of them, I humbly dedicate this work.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . .	iii
LIST OF FIGURES . . . . .	vii
ABSTRACT . . . . .	viii
CHAPTERS	
1 INTRODUCTION . . . . .	1
2 RELATED WORK . . . . .	4
3 OPTIMAL MULTICAST TREES . . . . .	10
3.1 Global Single Tree . . . . .	12
3.2 Multiple Trees . . . . .	16
4 OMT FOR LINEAR NETWORKS . . . . .	22
4.1 Single Group . . . . .	22
4.2 Multiple Groups . . . . .	23
4.3 Primary Destinations . . . . .	27
4.4 Minimizing the Real Cost of the OMT . . . . .	28
4.5 Multiple Trees . . . . .	28
5 OMT FOR RING NETWORKS . . . . .	38
5.1 Single Group . . . . .	38
5.2 Multiple Groups . . . . .	40
5.3 Primary Destinations . . . . .	45
5.4 Minimizing the Real Cost of the OMT . . . . .	45
5.5 Multiple Trees . . . . .	47
6 MULTICAST TREES FOR MESH NETWORKS . . . . .	53
6.1 Single Group . . . . .	53
6.2 Multiple Groups . . . . .	67
6.3 Primary Destinations . . . . .	69
6.4 Multiple Trees . . . . .	70

7	MULTICAST TREES FOR HYPERCUBE NETWORKS . . . . .	76
7.1	Single Group . . . . .	76
7.2	Multiple Groups . . . . .	85
7.3	Primary Destinations . . . . .	85
7.4	Multiple Trees . . . . .	86
8	CONCLUSION AND FUTURE WORK . . . . .	92
REFERENCES . . . . .		94
BIOGRAPHICAL SKETCH . . . . .		97

## LIST OF FIGURES

4.1	Form of each tree in a linear network . . . . .	31
4.2	Cost for Linear Network (N=64) . . . . .	35
4.3	Traffic for Linear Network (N=64) . . . . .	36
4.4	Primary Destinations for Linear Netwrok (N=64) . . . . .	37
5.1	Form of tree in a ring network . . . . .	41
5.2	Cost for Ring Network (N=64) . . . . .	50
5.3	Traffic for Ring Netwrok (N=64) . . . . .	51
5.4	Primary Destinations for Ring Netwrok (N=64) . . . . .	52
6.1	. . . . .	54
6.2	. . . . .	55
6.3	Optimal Algorithm vs Heuristic (N=64) . . . . .	66
6.4	. . . . .	69
6.5	Cost for Mesh Network (N=64) . . . . .	73
6.6	Traffic for Mesh Network (N=64) . . . . .	74
6.7	Primary Destinations for Mesh Network (N=64) . . . . .	75
7.1	Covered heuristic vs Our heuristic . . . . .	84

7.2	Cost for Hypercube Network (N=64) . . . . .	89
7.3	Traffic for Hypercube Network (N=64) . . . . .	90
7.4	Primary Destinations for Hypercube Network (N=64) . . . . .	91

Abstract of Dissertation Presented to the Graduate School  
of the University of Florida in Partial Fulfillment of the  
Requirements for the Degree Doctor of Philosophy

OPTIMAL MULTICAST TREES  
TO PROVIDE MESSAGE ORDERING

By

JAVIER E. CORDOVA

August 1993

Chairman: Dr. Yann-Hang Lee

Major Department: Computer and Information Sciences

A multicast group is a set of nodes that are the common destinations of the same group of messages. The source or sources may be nodes within the multicast group or may be other nodes in the network. Each message is addressed to all the elements of the group. Some applications require that the multicast messages be received in the same relative order by the nodes in the multicast groups. Our objective is to provide the multiple group ordering property for some known topologies (linear, ring, mesh, hypercubes) by constructing multicast trees. Two different graph models can be used to guarantee the message ordering properties. In the first one, a single global tree is built for all the nodes in overlapping groups, while in the second one, single trees are

built only for overlapping nodes, and on top of these, separate trees are built for each multicast group. The goal in both methods is to find multicast trees which minimize time and traffic.

## CHAPTER 1

### INTRODUCTION

A multicast group is a set of nodes that are the common destinations of the same group of messages. The source or sources may be nodes within the multicast group, or may be other nodes in the network. Each message is addressed to all the nodes of the group. Broadcasting is a special case of multicasting, where the nodes that make the group are all the nodes in the system.

There are many applications in which it is necessary that the nodes in a multicast group receive two or more messages in the same relative order. Suppose, for example, that nodes  $N_1$  and  $N_2$  receive, from the same source or from different sources, messages  $m_1$  and  $m_2$ . Nodes  $N_1$  and  $N_2$  should order the messages  $m_1$  and  $m_2$  in the same way. That is, the multicast protocol must guarantee that both nodes agree upon the order in which the messages will be processed. Consider for example a bank with two main computers. Each of these maintains a copy of the banking database, and they will process transactions from the branch offices. In this case, the two main computers may constitute a multicast group, and each branch office is a potential source. Now consider a deposit and a withdrawal to the same account. If the withdrawal is done first, an overdraft may occur and a penalty charged. If the deposit is done first there may be no penalty. Thus a different account balance may result if the two transactions are not executed in the same relative order.

Three different ordering properties can be required by a multicast protocol:

- single source ordering - in this case messages will be processed in the same relative order only if they are originated by a single source, and are destined to the same multicast group.
- multiple source ordering - in this case messages will be processed in the same relative order even when they originate from different sources, but are addressed to the same multicast group.
- multiple group ordering - this provides the general condition, in which we can have multiple sources, and the messages can be processed in the same relative order even when they are addressed to different but overlapping groups.

The banking example given above must satisfy property b). Now consider adding a second multicast group to distribute new software releases or new system tables (like defining overdraft penalty charges). This second multicast group consists of the two main computers and other development machines. It may be required that the messages still be processed in the same relative order by the intersection of the two multicast groups. In this case multiple group ordering must be provided.

It is easy to satisfy single source ordering. The source can assign a sequence number to every message, providing a total ordering of the messages. It is harder to deal with the other two properties. In the second chapter, existing algorithms to solve this problem are reviewed. The third chapter contains a description of our approach to attack the problem. Chapters four, five, six and seven describe the results we

have obtained for linear, ring, mesh and hypercube networks, respectively. The last chapter concludes our work, and points out directions for future research on this area.

## CHAPTER 2

### RELATED WORK

Kaashoek et al. [26] have proposed an atomic, reliable broadcast protocol for a broadcast medium based on a central site that assumes the role of a central sequencer. When a process wants to broadcast a message, it sends a point-to-point message to the central sequencer. The sequencer assigns the next sequence number to the message and broadcasts the message. In this way, the sequencer provides a global order for every broadcast message. All broadcasts are issued by the sequencer. The protocol handles missing messages in a simple manner, and with low overhead. The sequencer maintains a history buffer to keep track of the messages that have been received by every node. The nodes communicate this information to the sequencer in a header field every time they send a point-to-point message to the sequencer for a broadcast request. This way the sequencer maintains a table, indexed by node number, containing the information that node  $i$  has received at least the messages from 0 to  $T_i$ . The sequencer can compute the lowest value in the table and discard all messages up to that value, since it knows that all nodes have received such messages. The protocol requires two messages per broadcast in most cases. A major disadvantage of this method (and of any centralized method), however, is that the central site could become a bottleneck in the system.

Another centralized approach has been proposed by Chang and Maxemchuk [4]. As in the previous protocol, the approach is based on a central site that provides the global ordering of messages. The central site is identified with a token. However, in this protocol the role of central site, or token site, is rotated among all the nodes in the network. The source sends the broadcast message to all the nodes, and the token site acknowledges the message. The positive acknowledge contains a sequence number to provide the message ordering. On each acknowledgment the token is transferred to the next site. A site accepts the token only if it has received all the messages so far. This provides fault-tolerance to the protocol. Tokens are transferred and accepted in acknowledgments, so they in general do not generate additional messages if there are messages to be acknowledged. Depending on system utilization, however, they may generate three messages per broadcast: the broadcast message, the positive acknowledge sent by the token site, and the transfer of the token after each acknowledgment. Fault-tolerance is obtained by transferring the token. If  $L$  nodes accept the token after a message has been acknowledged, then the protocol can tolerate  $L$  processor failures. Long delays are introduced, however, before the messages can be processed.

Some distributed approaches have also been proposed. The first widely used method to establish order among events in a distributed system was proposed by Leslie Lamport [27]. In this approach, timestamps are assigned to every message at the source, and the messages are delivered in timestamp order. Using logical clocks in the sites a total ordering of the messages can be obtained. The major problem

with this kind of approach is that every process must know about the other process's operations. To guarantee the multiple source and multiple group ordering properties, when a node receives a message with its timestamp, before processing that message the node must check all nodes in the network (all of them are potential sources to the multicast group) for other messages not yet received with smaller timestamps addressed to the same multicast group or to an overlapping group. Only when the node is certain that it has received the message with the smaller timestamp can it proceed to process the message.

Birman and Joseph [3] have proposed a two-phase protocol to provide total ordering of messages. Each site maintains a priority queue associated with each process. When a node receives a message, it assigns a unique priority number to it higher than any other given to that process before, marks the message as *undeliverable*, and sends this priority number to the sender of the message. The sender computes the maximum priority number from all the priorities received from all active destinations, and sends this value back to the destinations. These change their initial priority number to the new one, tag the message as *deliverable*, and re-arrange their priority queues. A message at the front of the queue whose tag is *deliverable* is then delivered. Their protocol is mainly concerned with fault tolerance, and hence they may have a large message overhead.

A different approach is followed by Garcia-Molina and Spauster [13]. For a given set of multicast groups with overlapping nodes, a single tree is generated, called a propagation graph. The graph indicates the path that each message must follow

to get to all the destination nodes. The sender sends the message to a *primary destination* of the multicast group, which is the smallest common ancestor to all the nodes in the group in the propagation graph. Instead of ordering the messages at a single central site, they are ordered by the collection of nodes in the tree by merging messages destined for different groups. Nodes that are in the intersections of the multicast groups are selected to be the intermediary nodes in the graph.

One technique that has been applied to provide multicasting in computer networks is single-tree and multiple-tree forwarding. In this technique, single or multiple spanning trees are constructed that span the nodes in the multicast group. Source-base forwarding may be used to construct such a tree for each node in the system. These trees can minimize both delay and bandwidth. The algorithms proposed, however, generally do not consider multiple groups.[11]

Minimum spanning trees and shortest path trees have been proposed to build trees for broadcasting and multicasting. Minimum spanning trees [9] minimize the total branch cost for a broadcast. For multicasting, minimum Steiner trees must be constructed to minimize the total cost of a multicast. However, finding minimal Steiner trees has been shown to be an NP-Complete problem [25]. In any case, these trees in general do not minimize message delay. Shortest path trees [37] minimize the distance from the root to any other node in the tree. Wall [37] discusses several ways of computing "a center" of a network and constructing its shortest path tree, so that the trees constructed have good delay properties.

Lan et al. [28] considered the problem of constructing shortest path trees and at the same time minimizing traffic for multicasting in hypercube networks. Esfahanian and Choi [6] have shown that building optimal multicast trees is an NP-Complete problem for hypercubes. They developed a heuristic algorithm, taking as the root of the tree the source of the multicast message. This approach leads to a multicast tree for each different source, which is inappropriate for message ordering.

The Steiner tree problem is closely related to the Optimal Multicast Tree problem (OMT). An OMT for a multicast group is defined as a tree that minimizes the maximum distance from the root to the farthest element from the root in the multicast group that has minimum traffic, that is, minimum number of links. A minimum Steiner tree minimizes only the traffic parameter. In general, optimal multicast trees do not minimize traffic, i.e., they are not minimum Steiner trees. Garey and Johnson [15] have shown that finding minimum Steiner trees for graphs with rectilinear distances is an NP-Complete problem. This has lead us to conjecture that building an OMT for mesh networks, which use rectilinear links, is also NP-Complete. The problem of building an *OMT* in a grid for a multicast group  $D$  given the root is equivalent to the problem of finding a rectilinear directed Steiner tree for  $D$  in a grid in which all the links point away from the root node. A special case of this problem (in which all the nodes lie in the first quadrant of  $E^2$  and the directed tree is rooted at the origin) has been known in the literature as the *Rectilinear Steiner Arborescence* problem (RSA). Rao et. al [32] have proposed a heuristic algorithm for this problem that has time complexity of  $O(|D| \log |D|)$ . They proved that

the traffic generated by the heuristic has an upper bound of twice of that generated by any optimal algorithm. They have proposed an extension of this algorithm that solves the general case in which the nodes in  $D$  lie anywhere in the plane that runs with  $O(|D|^3 \log |D|)$  time complexity. We propose in this dissertation a similar heuristic with same upper bound that also has time complexity  $O(|D| \log |D|)$  but solves this general case in which the nodes are allowed to be anywhere in  $E^2$ .

## CHAPTER 3

### OPTIMAL MULTICAST TREES

The objective of our work is to present a technique to guarantee the message ordering properties for several known topologies: linear, ring, mesh and hypercubes. Multicast messages will be ordered by a collection of nodes structured in a message propagation graph. The graph indicates the flow of the messages through the network.

An approach that has been described to handle multicasting is the single-tree forwarding technique. In this technique a multicast tree is constructed that spans the nodes in the multicast group. Each node sends message copies along the branches of the tree. Given a single multicast group, a multicast tree can guarantee that all the nodes in the group receive the messages in the same relative order. The root of the tree decides the message ordering. With multiple groups, single trees are needed that span the nodes in each multicast group. To guarantee the multiple group ordering property, a single tree (or subtree) can be constructed for overlapping nodes in the multicast groups. Two different graph models could be used to guarantee the message ordering properties. In the first one, a single global tree is built for all the nodes in the overlapping groups, while in the second one, single trees are built only for overlapping nodes, and on top of this subtrees, separate trees are built for each multicast group. In any of these graphs, the source sends a multicast message to the closest ancestor

of all the nodes in the multicast group. This common ancestor is called the *primary destination* of the multicast group.

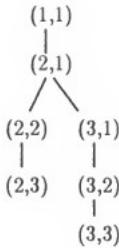
It is assumed that any site can be the source of messages to a particular destination group. When a node wants to send a message to a particular group, a path from the source to the primary destination must be determined. In the process of constructing multicast trees, the source of the messages is not taken into consideration, since it is assumed that the sources to the multicast groups are uniformly distributed.

Two parameters must be considered to construct a multicast tree: time and traffic. The parameter time is a measure of the time required to send a message from the primary destination of a multicast group to the last node in the group that receives the message. The time to send a message from one node to another is given by the *distance* between the two nodes. The *cost* of the multicast is associated with the time required to complete the multicast, and is determined by the distance between the primary destination and the node in the group farthest from the primary destination. A message can be forwarded in parallel through disjoint paths in the tree. The traffic parameter is given by the number of links needed to multicast the message from the primary destination to every element of the group through the branches of the multicast tree. The messages should arrive as fast as possible to all their destinations, using as few links as possible to achieve that. These two parameters are not independent. For example, consider a mesh network in which the nodes  $(2, 1), (3, 1), (3, 2), (3, 3), (2, 3)$  constitute a multicast group and the root of the tree is given by the node  $(1, 1)$ . The multicast tree that minimizes the traffic generated

is given by

$$(1,1) \rightarrow (2,1) \rightarrow (3,1) \rightarrow (3,2) \rightarrow (3,3) \rightarrow (2,3)$$

This tree generates a traffic of 5 units. However, the cost (or time) is also 5, while the tree given by

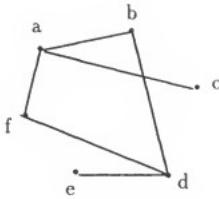


generates a traffic of 6, but its cost is only 4 units.

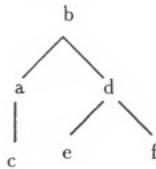
An *optimal multicast tree* (OMT) is a multicast tree with a minimum cost that has the lowest traffic.

### 3.1 Global Single Tree

Suppose we partition the groups so that no group in a partition intersects a group in another partition, and we form as many partitions as we can with this property. A single tree can be constructed that spans the nodes in each partition. For example, consider the following multicast groups:  $A = \{a, b, c, d\}$ ,  $B = \{d, e, f\}$ ,  $C = \{a, b, f\}$ , where the capital letters A,B,C represent the multicast groups, and the lower case letters represent the nodes. Suppose that the connections of the network are given by the following graph:



Then an optimal multicast tree is given by:



Node  $b$  is the primary destination for groups A and C, and node  $d$  is the primary destination for group B. Two different messages sent to groups A and C are ordered by node  $b$ .

*Lemma 3.1* *The multiple group ordering property is satisfied under the global single tree model.*

**Proof:**

We need to show that any two nodes  $a$  and  $b$  belonging to two or more groups will receive the messages addressed to these groups in the same relative order. Since all of these groups contain nodes  $a$  and  $b$ , it must be the case that their primary destinations lie on or above  $a$  and  $b$  in the multicast tree. Let  $P$  be the lowest one of these primary destinations, i.e., the closest common ancestor to  $a$  and  $b$ . Every message addressed to any of the groups that contain nodes  $a$  and  $b$  has to go through

node  $P$  before reaching nodes  $a$  and  $b$ . The order of the messages for the nodes in the groups that contain both  $a$  and  $b$  can thus be decided by node  $P$ , and nodes  $a$  and  $b$  will agree in that order.  $\square$

Essentially, two steps are required to construct the multicast tree for a partition. First, an appropriate root must be found. Once an optimal root is computed, we must determine the best routing strategy to cover every node in the multicast groups, and determine the primary destinations.

Consider first the problem of finding an optimal root. We assume that sources are uniformly distributed among the nodes of the network. We also assume that the probability that an arbitrary message is addressed to a particular destination group is the same for every group. We want to select the root such that the sum of all distances from the root to the elements in each group farthest from the root is minimized. The problem can be formulated as follows.

Let  $N$  be the number of nodes in the network. The set of nodes will be represented by the set  $\{0, 1, \dots, N - 1\}$ . Let  $d(x, y)$  be the length of a shortest distance between nodes  $x$  and  $y$ . Let  $n_{ij}$  be the  $j$ th node of the  $i$ th group in a multicast partition.

The optimal root of the  $OMT$  is defined as the node that produces the minimal cost. The cost for a root node  $i$  is defined by:

$$c_i = \sum_{j=1}^p \max_{1 \leq k \leq m_j} d(pd_i, n_{jk})$$

where  $p$  is the number of multicast groups,  $pd_i$  is the primary destination for group  $i$ , and  $m_j$  is the number of nodes in the  $j$ th multicast group. The root node with

minimal cost  $c_m$ , is thus

$$c_m = \min_{0 \leq i \leq N-1} c_i$$

A problem here is that the primary destinations are not known before the tree is built, and even before the root has been found. In some cases it may be necessary to find the root that minimizes the sum of the distances from the root to the farthest element in each multicast group, and not necessarily the sum of the corresponding distances from the primary destinations. In this case we would be minimizing the maximum potential cost of the multicast tree, and not the actual cost of the tree. The cost of the tree is then given by

$$c_i = \sum_{j=1}^p \max_{1 \leq k \leq m_j} d(i, n_{jk})$$

A naive way of computing the cost of every node requires time complexity  $O(Nm)$ , where  $m = \sum_{j=1}^p m_j$ , assuming that the distances between each pair of nodes can be computed in constant time.

Once we have the root, the rest of the tree must be determined. Routes must be found from the root to each node in the multicast groups. Determining the best routing for linear and ring networks is simple and can be done in time proportional to  $N$ . For mesh and hypercubes networks the problem is much harder. Shortest-path trees will be considered for these networks. A *shortest-path tree* is a tree that spans all the elements of the destination lists, with the property that the path from the

root to every node in the tree is a shortest path in the original network. The routing problem can be formulated as follows.

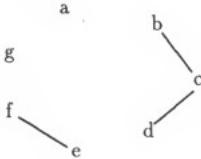
Let  $G$  be the graph determined by the network topology, and let  $d_G(x, y)$  be the length of a shortest path in graph  $G$  from node  $x$  to node  $y$ . We want to construct a tree  $T$  of root node  $R$  with the following properties:

- $V(T) \subseteq V(G)$
- the sum of the maximum distances from the root node  $R$  to the element in each multicast group farthest from  $R$  is minimized
- $d_T(x, R) = d_G(x, R) \forall x \in T$
- no tree exists satisfying the three properties above that has fewer nodes than this one.

### 3.2 Multiple Trees

The multiple group ordering property can be satisfied without constructing a single global tree, as long as overlapping nodes belonging to two or more groups belong to common subtrees in the trees for each of these multicast groups. Given the multicast groups, define graph  $G' = (V', E')$  such that  $V'$  is the set of nodes in the groups, and for  $v_1, v_2 \in G'$ , the edge  $(v_1, v_2) \in E'$  iff  $\{v_1, v_2\} \subseteq g_i \cap g_j$  for any two groups  $g_i$  and  $g_j$ . That is, two nodes are connected in  $G'$  if and only if they both belong to the intersection of at least two groups. If  $(v_1, v_2) \in E'$  then  $v_1$  and  $v_2$  are overlapping nodes, so they must receive messages addressed to groups  $i$  and  $j$  in the same relative order. They will belong to a common subtree of the multicast trees for

these groups. In fact, all *connected components* of  $G'$  will be in common subtrees. For example, consider the following groups:  $A=\{a,b,c,g\}$ ,  $B=\{b,c,d\}$ ,  $C=\{c,d,e,f\}$ , and  $D=\{e,f\}$ . Then the graph  $G'$  is given by:



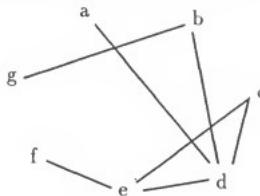
A multicast tree  $T_i$  has to be constructed for every group  $i$ . The next step is to build subtrees that span the nodes in each connected component of  $G'$ , and then connect these subtrees such that the following condition is satisfied: a subtree for a connected component would be a subtree of  $T_i$  if the connected component contains at least two elements of group  $i$ . In the previous example, the subtree with the connected component  $\{b,c,d\}$  must be a subtree of the multicast tree for groups A, B and C, and the one with  $\{e,f\}$  must be a subtree for groups C and D.

Once the subtrees have been built, a multicast tree  $T_i$  is built for each group  $i$  in the following way:

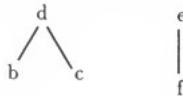
1. First we need to determine which subtrees  $S_j$  need to be part of  $T_i$ . Every leaf in  $S_j$  must belong to group  $i$ . The nodes and links in the subtree that do not have descendants in group  $i$  are not included in the subtree for  $T_i$ . Let  $A$  be the set of nodes to be connected in step 3. Add the root of  $S_j$  to set  $A$ , and assign a depth to this node equal to the depth of  $S_j$ . In case this node was assigned a previous depth  $x > 0$ , then assign to it  $\max\{x, \text{depth of } S_j\}$ .

2. Consider the nodes  $j$  in group  $i$  that did not belong to any connected component. If node  $j$  can be covered by any of the subtrees found in the previous step without increasing the cost of the subtree, then consider this node as covered by the subtree (this may affect the traffic generated by the subtree, but not its cost). This node is inserted below the closest node to it already in the subtree that keeps the subtree a shortest-path tree. The longest possible path already in the tree is used as a shortest-path from the root to node  $j$ . This helps in reducing the traffic. The other nodes that cannot be inserted without increasing the cost of the subtree are added to  $A$  and a depth of 0 is assigned to them.
3. Connect the nodes in set  $A$ .

Consider the multicast groups given above, together with the following network:

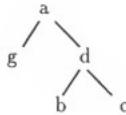


The subtrees generated for the two connected components may be as follows:



Now we need to determine which of these subtrees (completely or partially) must belong to the multicast tree for each group. These subtrees must be connected together, and to the extra nodes in the group that did not belong to any connected component.

For group A, for example, only the first subtree must be part of its multicast tree. Nodes  $a$  and  $g$  must be connected to this subtree. A final multicast tree for this group could be

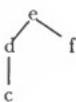


For groups B,C and D the following trees could be obtained:

B



C



D



The primary destination for group A is node  $a$ , for group B is  $d$ , and for groups C and D is node  $e$ .

*Lemma 3.2* *The multiple group ordering property is satisfied under the multiple trees model.*

#### Proof:

We need to show that two nodes  $a$  and  $b$  belonging to two or more groups will receive the messages addressed to these groups in the same relative order. Suppose that nodes  $a$  and  $b$  belong together to the groups in set  $C = \{g_i \mid a, b \in g_i\}$ . Nodes  $a$  and  $b$  will be connected by a direct link in graph  $G'$ . Therefore, they will belong to a common subtree, and this subtree will be part of the multicast tree for each of the groups in  $C$ . Let  $P$  be the root of this subtree. Every message addressed to the

groups in  $C$  must go through node  $P$  before reaching nodes  $a$  and  $b$ . Thus  $P$  can provide the ordering of the messages for the nodes belonging to the groups in  $C$ .  $\square$

Algorithms to construct single trees given a set of nodes are given in this dissertation for linear, ring, mesh and hypercube networks. These algorithms will be used to generate the subtrees for each connected component of  $G'$  when we want to construct multiple trees. They are used as well to connect the necessary subtrees together and the rest of the nodes of each group that did not belong to any connected component.

In the single global tree case, an optimal root for the multicast tree is determined given the members of the multicast groups. Here, an optimal root must be found given a set of subtrees and a set of single nodes. The subtrees will be connected via their roots. To compute the optimal root in this case, the previous algorithms would be modified to take into consideration the depth of the subtrees to be connected. Each root of a subtree is assigned a depth equal to the depth of the subtree it represents, and the single nodes that did not belong to any intersections are assigned a depth of 0. The whole idea is again to minimize the cost of the multicast for each group. The structure of the subtrees generated by the connected components would remain intact to guarantee the multiple group ordering property. Once the optimal root is found, the routing from this root to the nodes connected is determined. The links used in these routes may include some links already used in the subtrees. Thus it may happen that a message has to go through one or more nodes more than once until the multicast is completed. This fact tends to increase the amount of traffic generated as compared to the single global tree solution, but the cost of the multicast

would be reduced, and much more variability in the primary destinations would be obtained.

In the following chapters, the two graph models to construct multicast trees are discussed in more detail for each of the network topologies considered, and their performance evaluation compared in terms of the costs of the multicast, the amount of traffic generated, and in terms of the percentage of multicast groups that have the same primary destination.

## CHAPTER 4 OMT FOR LINEAR NETWORKS

### 4.1 Single Group

Let the multicast group be composed by the nodes  $n_1, \dots, n_m$ . Let  $l = \min_{1 \leq i \leq m} n_i$ , and  $h = \max_{1 \leq i \leq m} n_i$ . Assuming that the multicast group is sorted in increasing order, then  $l = n_1$ , and  $h = n_m$ . The optimal root  $r$  is computed by  $r = \lfloor (l + h)/2 \rfloor$

Clearly,  $r$  is the midpoint of the interval from  $n_l$  to  $n_h$ , and  $c_r = \min_{0 \leq i \leq N-1} c_i$ .

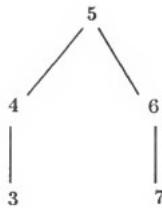
The optimal root can be found in  $O(1)$ . In a linear network, constructing the multicast tree once we have the optimal root is trivial. There is only one path from each node to every other node, so there is no room to minimize traffic. This unique path has to be taken in the multicast tree, and only one message is sent by every node in the tree to each child.<sup>1</sup>

*Example 4.1 :*

Let  $N = 8$ , and the multicast group be composed of the nodes 3, 6, and 7. Then the optimal root is node 5, and the optimal multicast tree is:

---

<sup>1</sup>Actually, every node except the root has only one child in the tree.



If a node has to multicast a message to this group, it sends the message to node 5 through a shortest path, and the message gets propagated down the tree to the nodes in the destination list. In this case, we say that the cost of the multicast is 2, which is the cost of the root.  $\square$

#### 4.2 Multiple Groups

Let  $p$  be the number of multicast groups. Let  $l_i = \min_{1 \leq j \leq m} n_{ij}$ , and  $h_i = \max_{1 \leq j \leq m} n_{ij}$ . Compute,  $\forall 1 \leq i \leq p$ ,  $r_i = (l_i + h_i)/2$ . Each of this  $p$  values represent the midpoint of each group. The following algorithm finds the optimal root. Note that the values of the  $r'_i$ 's are either integers or lie exactly between two integers.

#### Algorithm 4.1 Linear-Multiple Groups

```

begin
  for i=1 to p do  $r_i = (l_i + h_i)/2$ ;
  sort the  $r'_i$ 's;
  if  $p$  is odd then
     $m = (p + 1)/2$ ;
    if  $r_m$  is an integer then  $root = r_m$ 
    else
      let  $nl$ =number of  $r'_i$ 's strictly less than  $r_m$ ;
      let  $nh$ =number of  $r'_i$ 's strictly higher than  $r_m$ ;
      if  $nl \leq nh$  then  $root = \lceil r_m \rceil$ 
      else  $root = \lfloor r_m \rfloor$ 
  else

```

```

 $m = p/2;$ 
if  $r_m$  is an integer then  $root = r_m$ 
else if  $r_m < r_{m+1}$  then  $root = \lceil r_m \rceil$ 
else
    let  $nl$  and  $nh$  be as before;
    if  $nl \leq nh$  then  $root = \lceil r_m \rceil$ 
    else  $root = \lfloor r_m \rfloor$ 
end;

```

### Proof of correctness:

There are essentially 4 different cases we need to consider to prove that the optimal root is computed by this algorithm. We will show that for all of them the computed value is correct.

- *Case 1:* In this case,  $p$  is odd and  $r_m$  is an integer.  $r_m$  is exactly the midpoint in the ordered list  $r_1, \dots, r_p$ . Consider the cost of a node  $x$ , such that  $x + z = r_m$ ,  $z > 0$ . The number of groups with midpoint  $r_i \leq x$  is less than 50% of the number of groups (less than  $p/2$ ). For this groups the cost of choosing  $x$  as the root will decrease by at most  $z$  units, as compared to the cost of  $r_m$ , while for more than 50% of the groups the cost will increase by  $z$ . The same argument applies for the case in which  $x > r_m$ . Thus  $c_{r_m}$  is necessarily optimal.
- *Case 2:*  $p$  is odd and  $r_m$  is not an integer. For this case, we have computed the value ( $r_m$ ) for which the cost is optimized, but this value does not represent a node, since it is not an integer. With a similar proof as the previous case, we can show that the optimal root would be one of the integers closest to  $r_m$ . The minimal cost would be  $\min(c_{\lfloor r_m \rfloor}, c_{\lceil r_m \rceil})$ . It is not necessary, however, to compute these two values to find the optimal solution. We can use the values

of  $r_1, \dots, r_p$ . Let  $nl$ ,  $ne$  and  $nh$  be the number of multicast groups that have midpoints less than, equal to, or greater than  $r_m$ , respectively. If  $\lceil r_m \rceil$  is chosen as the root, then  $nl$  groups will increase by 0.5 units each the total cost, while  $nh$  groups will decrease it by 0.5 each. The inverse holds if  $\lfloor r_m \rfloor$  is selected. The groups with midpoints equal to  $r_m$  will increase the cost by 0.5 anyway. Thus,

$$c_{\lceil r_m \rceil} = .5nh - .5nl + .5ne + c_{r_m}$$

and

$$c_{\lfloor r_m \rfloor} = .5nl - .5nh + .5ne + c_{r_m}$$

Subtracting these two equations, we get that

$$c_{\lceil r_m \rceil} - c_{\lfloor r_m \rfloor} = nh - nl$$

and therefore,

$$c_{\lceil r_m \rceil} \geq c_{\lfloor r_m \rfloor} \iff nh \geq nl$$

Thus the algorithm computes the optimal solution.

- *Case 3:*  $p$  is even, and there exists at least one integer  $x$  such that  $r_m \leq x \leq r_{m+1}$ . It can be shown that every integer between  $r_m$  and  $r_{m+1}$  is an optimal solution when  $p$  is even. We will show that in particular,  $\lceil r_m \rceil$  is optimal. The proof is similar to the proof of case 1.

Consider first the cost of a node  $y$  such that  $y < \lceil r_m \rceil$ . Note that  $y < r_m$ , and thus, as in case 1, the number of groups with mean  $r_i \leq r_m$  is more than 50%, or  $p/2$ . For these groups the cost of choosing  $y$  will decrease by at most  $\lceil r_m - y \rceil$ , while the cost for more than 50% of the groups will decrease by exactly this amount, increasing the overall cost.

For  $y > \lceil r_m \rceil$ , the number of groups with midpoint  $r_i \leq y$  is at least 50% (may be more, if  $r_{m+1} = r_m$ ). Thus, the cost of selecting  $y$  will increase by at least half the number of groups, and never can decrease by that much, so  $c_y$  could never be lower than  $c_{r_m}$ . Therefore,  $\lceil r_m \rceil$  is optimal.

- *Case 4:*  $p$  is even,  $r_m$  is not an integer, and there is no integer between  $r_m$  and  $r_{m+1}$ . This implies that  $r_m = r_{m+1}$ . This case is essentially the same as the one discussed in case 1.  $\square$

The algorithm computes the optimal root with time complexity  $O(p \log p)$ . Essentially, the algorithm computes the *median* of the group midpoints.

*Example 4.2 :*

Let  $N = 8$ ,  $p = 3$ , and the multicast groups be 0,4,5, and 2,3,4, and 5,7. Then  $r_1 = 2.5$ ,  $r_2 = 3$ , and  $r_3 = 6$ . The optimal root is 3, and  $c_3 = 3 + 1 + 4 = 8$ . The cost of the nodes here is:

node	cost
0	$16=5+4+7$
1	$13=4+3+6$
2	$10=3+2+5$
3	$8=3+1+4$
4	$9=4+2+3$
5	$10=5+3+2$
6	$11=6+4+1$
7	$14=7+5+2$
8	$17=8+6+3$

### 4.3 Primary Destinations

The following algorithm computes the primary destinations of  $p$  multiple groups for the linear case in  $O(p)$ . The root of the multicast tree is  $R$ . The primary destination for group  $i$  is stored in  $PD_i$ .

#### Algorithm 4.2 Primary destinations for linear multicast trees

```

begin
  for i=1 to p do
    if ( $l_i \leq R$ ) and ( $h_i \geq R$ ) then  $PD_i = R$ 
    else if  $l_i \leq R$  then  $PD_i = h_i$ 
    else  $PD_i = l_i$ 
  end;

```

#### 4.4 Minimizing the Real Cost of the OMT

In the previous sections, we have minimized the sum of the distances from the root to the node in each group farthest from the root. The real cost of a multicast tree is, however, the sum of the distances from the *primary destination* of each group to the node farther from it in that group. The following algorithm, which is  $O(Np)$ , finds the root for the optimal multicast tree in this sense:

Algorithm 4.3 Optimal root in terms of the primary destinations

```

begin
  for r=1 to N-2 do  $c_r = 0$ ;
  for r=1 to N-2 do
    for i=1 to p do
      if ( $h_i \leq r$ ) or ( $l_i \geq r$ ) then  $c_r = c_r + h_i - l_i$ 
      else  $c_r = c_r + \max\{r - l_i, h_i - r\}$ ;
   $r = \min_{1 \leq r \leq N-2} c_r$ 
end;
```

The primary destinations can be computed as before in  $O(p)$  time.

#### 4.5 Multiple Trees

As mentioned before, the multiple group ordering property can be satisfied without constructing a single global tree, as long as overlapping nodes belonging to two or more groups belong to common subtrees in the trees for each of these multicast groups. Given a series of multicast groups, the graph  $G(V, E)$  is defined as follows:  $V$  is the set of nodes in the multicast groups, and for any two nodes  $u, v, (u, v) \in E$  iff  $\{u, v\} \subseteq g_i \cap g_j$  for any two groups  $g_i$  and  $g_j$ . That is, two nodes are connected in  $G$  if and only if they both belong to the intersection of at least two groups. Two nodes connected by an edge in  $G$  belong to at least two groups  $i$  and  $j$ . These two nodes

must receive messages addressed to groups  $i$  and  $j$  in the same relative order. Thus, a subtree containing both of these nodes must be common to the multicast trees built for the two groups. But two nodes need not be directly connected by an edge to require that they belong to a common subtree in the trees for two or more groups. For example, consider the following groups:  $A = \{a, b, c\}$ ,  $B = \{a, b\}$ ,  $C = \{b, c\}$ . In this case, the graph  $G$  does not include a direct link between nodes  $a$  and  $c$ . However, to guarantee the multiple group ordering property it is required to construct a subtree containing all the three nodes. To guarantee the ordering property, subtrees for each connected component of  $G$  are built. Each of these subtrees, in part or completely, will belong to the multicast tree of two or more groups. The subtrees are built in the same way as we built multicast trees for single groups. Once the subtrees have been built, the three steps described in chapter 3 to construct the multicast tree  $T_i$  for group  $i$  are executed. These steps were:

1. First we need to determine which subtrees  $S_j$  need to be part of  $T_i$ . Every leaf in  $S_j$  must belong to group  $i$ . The nodes and links in the subtree that do not have descendants in group  $i$  are not included in the subtree for  $T_i$ . Let  $A$  be the set of nodes to be connected in step 3. Add the root of  $S_j$  to set  $A$ , and assign a depth to this node equal to the depth of  $S_j$ . In case this node was assigned a previous depth  $x > 0$ , then assign to it  $\max\{x, \text{depth of } S_j\}$ .
2. Consider the nodes  $j$  in group  $i$  that did not belong to any connected component. If node  $j$  can be covered by any of the subtrees found in the previous step without increasing the cost of the subtree, then consider this node as

covered by the subtree (this may affect the traffic generated by the subtree, but not its cost). This node is inserted below the closest node to it already in the subtree that keeps the subtree a shortest-path tree. The longest possible path already in the tree is used as a shortest-path from the root to node  $j$ . This helps in reducing the traffic. The other nodes that cannot be inserted without increasing the cost of the subtree are added to  $A$  and a depth of 0 is assigned to them.

### 3. Connect the nodes in set $A$ .

We discuss now how the nodes in  $A$  are connected for linear networks. Note that now the cost of node  $i$  is given by  $c_i = \max_{j \in D} d(i, j) + \text{depth}(j)$ . First we find the smallest node  $a \in A$  such that  $\text{depth}(a) \geq a - x + \text{depth}(x)$  for any node  $x \in A$  with  $x \leq a$ . Similarly, node  $b \in A$  is the largest node in  $A$  such that  $\text{depth}(b) \geq y - b + \text{depth}(y)$  for any node  $y \in A$  with  $y \geq b$ . Note that it may happen that  $a = b$ . Let  $r' = (a + b + \text{depth}(b) - \text{depth}(a))/2$ . If  $r' \leq a$  then  $r = a$ . Else if  $r' \geq b$  then  $r = b$ . Else  $r = r'$ .

Lemma 4.1  $r$  is the optimal root.

**Proof:**

Assume, for simplicity, that  $r'$  is an integer. If  $r' \leq a$  or  $r' \geq b$  then  $r$  is clearly the optimal root. Otherwise  $r$  lies exactly in the middle of  $a$  and  $b + \text{depth}(b) - \text{depth}(a)$ , and therefore  $r - a + \text{depth}(a) = b - r + \text{depth}(b) = c_r$ . Let  $x$  be an optimal root. If  $x < r$  then  $c_x > b - r + \text{depth}(b) = c_r$ , which is a contradiction. Similarly, if  $x > r$ , then  $c_x > c_r$ . Thus  $x = r$ , and hence the proof is complete.  $\square$

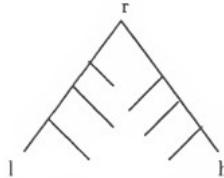


Figure 4.1. Form of each tree in a linear network

The final tree  $T_i$  for group  $i$  must have this form given in figure 4.1, where  $r$  is the optimal root, and  $l$  and  $h$  are the lowest and highest nodes in group  $i$ , respectively.

Some of the nodes in the path from  $r$  to  $l$  may have a linear portion to their right side. Similarly, some of the nodes in the path from  $r$  to  $h$  may have a linear portion to their left side. These nodes were roots of the subtrees built for the connected components of graph  $G$ . The nodes in the other side of these subtrees can be reached by the paths from  $r$  to  $l$  and  $h$ , preserving the structure of the subtrees. This helps reduce the traffic generated by the multicasts.

Example 4.3 :

Let the multicast groups be given by:  $A = \{0, 2, 3\}$ ,  $B = \{2, 3, 4, 5, 7\}$ ,  $C = \{3, 5\}$ ,  $D = \{4, 6, 7, 8\}$ .

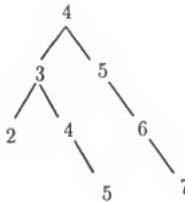
The connected components are given by:  $\{2, 3, 5\}$  and  $\{4, 7\}$ . The subtrees for these two connected components are, respectively:



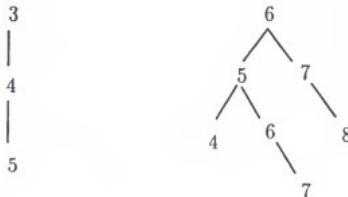
For group *A*, only the first subtree must be part of its multicast tree. We need to connect node 0 (depth=0) and node 3 (depth=1). The optimal root is node 2, and the multicast tree is given by:



For group *B* we need to connect both subtrees, having both roots 3 and 5 a cost of 2. The tree for group *B* is:



For group *C* only a segment of the first subtree is needed. For group *D*, only the second subtree is needed. Also, when nodes 6 and 8 are considered in step 3, node 6 is discarded since it is covered by the subtree, and node 8 need to be connected with node 5. The trees for groups *C* and *D* are shown below:



- The primary destinations for groups  $A, B, C, D$  are respectively, nodes 2, 4, 3, and 6. In this case the node that is primary destination a maximum number of times is primary destination for 25% of the multicast groups.  $\square$

In the previous example, the cost to complete the multicast is given by  $2+3+2+3=10$ , while the traffic generated is  $4+7+2+6=19$ . Using the single global tree approach for this example we would have obtained the multicast given by:



with a cost of  $3+3+1+4=11$ , and a traffic of  $3+5+2+4=14$ . Node 4 is now primary destination for 75% of the groups.

The following graphs illustrate our results from simulations ran for a linear network with 64 nodes. To increase the number of nodes in the intersections of the multicast groups, we assumed the existence of "hot nodes" in the network. In our simulations, the first 25% of the nodes in the network have a fixed probability of 0.2 of being included in a multicast group, while the rest have only probability of 0.1. Figure 4.2 shows the cost of the multicasts (assuming that one multicast message is addressed to each multicast group) for the randomly generated multicast groups. The graph shows that the cost of the multicasts is lower for the multiple trees model than

for the single global tree model. Figure 4.3 shows that the traffic generated is higher for the multiple trees approach. As mentioned before, this result was expected, since in this model the same message may be required to be sent over the same link (in the same or in opposite directions) more than once. The last example given above illustrates this fact. Thus the traffic was expected to be higher in the multiple trees approach to guarantee the multiple group ordering property. The third graph shows what is probably the most significant advantage of the multiple trees model: much higher variability in the primary destinations. In the single global tree model, the same node (the root of the tree) must be the primary destination for most of the multicast groups. In that case we do not deviate too much from a centralized solution to the multiple group ordering problem, which has the potential disadvantage of developing a bottleneck in the root. In the multiple trees approach the function of being a primary destination is much more distributed among different nodes in the network.

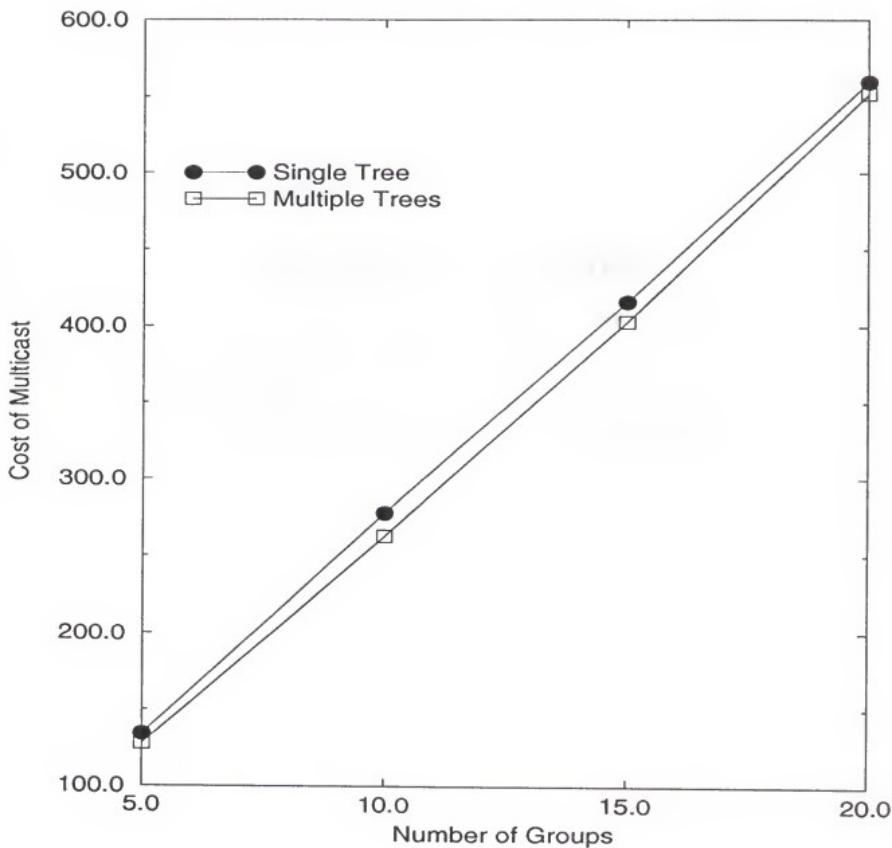


Figure 4.2. Cost for Linear Network ( $N=64$ )

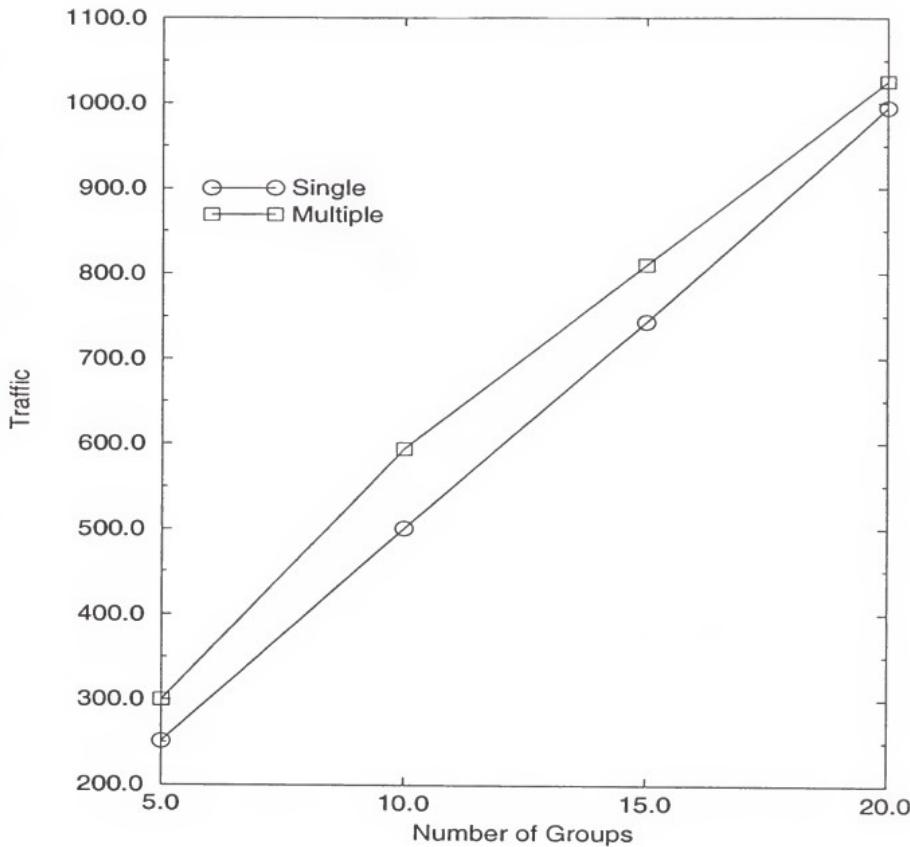


Figure 4.3. Traffic for Linear Network ( $N=64$ )

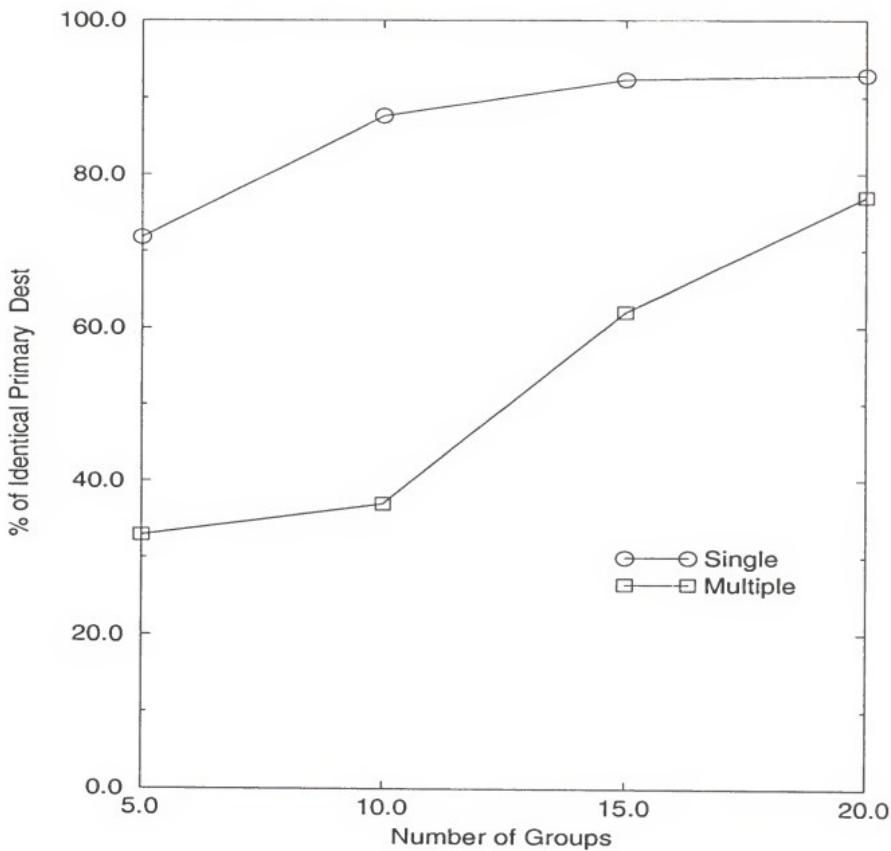


Figure 4.4. Primary Destinations for Linear Netwrok (N=64)

## CHAPTER 5 OMT FOR RING NETWORKS

### 5.1 Single Group

Let  $R_N$  be the ring network with nodes  $0, 1, 2, \dots, N-1$ . Let  $D$  be the destination list. A path from node  $x$  to node  $y$  is denoted by  $(x, y)$ . Nodes  $x$  and  $y$  are the *endpoints* of this path. All other nodes in the path are called *interior points*. The set of interior points for a path  $P$  is denoted by  $I(P)$ . For every pair of nodes  $x$  and  $y$  there are two paths in  $R_N$  that connects them. When it becomes necessary to differentiate between them, extra nodes in the path would be added to denote it, like for example,  $(x, x+1, \dots, y)$  is the path from  $x$  to  $y$  that goes through node  $x+1$ . For simplicity, sometimes the set  $D$  is represented without brackets and without commas between its elements. For example,  $D = 024$  denotes the multicast group composed of the nodes 0, 2 and 4. The *complement of a path*  $P$   $(x, x+1, \dots, y)$  is defined as the path  $(x, x-1, \dots, y)$ , and viceversa. It is denoted by  $\overline{P}$ . Here, additions are modulo  $N$ .

Let  $P$  be the longest path in the ring such that if  $x \in I(P)$  then  $x \notin D$ .

Lemma 5.1 *The optimal root is the midpoint of the complement of path  $P$ .*

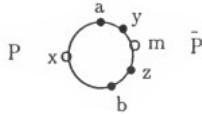
**Proof:**

Let  $m$  be the midpoint of  $\overline{P}$ . Let  $a$  and  $b$  be the endpoints of  $P$ . Clearly,  $a$  and  $b$  are

elements of  $D$ . Assume that if the number of nodes in path  $\overline{P}$  is even, then we define  $m$  to be the node that is closer to  $b$ . Thus,  $c_m = d(m, a) \leq d(m, b) + 1$ . We need to show that  $m$  is an optimal solution, that is, for any other node  $x$  in the network,  $c_m \leq c_x$ . It should be clear that any node in  $\overline{P}$  satisfies this condition. We need to show that it is also true for nodes in  $I(P)$ . Let  $x \in I(P)$ .

*Case 1:*  $m \in D$ . Clearly,  $c_m = d(m, a) \leq d(m, x)$ . Therefore,  $c_m \leq c_x$ , and we are done. Note that  $a$  is the node in  $D$  farthest from  $m$ .

*Case 2:*  $m \notin D$ . Assume, by contradiction, that  $c_x < c_m$ . Let  $y$  and  $z$  be the elements in  $D$  closest to  $m$  such that  $y$  is closer to  $a$  and  $z$  is closer to  $b$ . Note that it may be the case that  $a = y$  or  $b = z$ . See the figure below to locate the nodes we have defined so far.



We have that

$$d(m, a) = d(m, y) + d(a, y)$$

$$d(x, y) = d(x, a) + d(a, y)$$

Subtracting the above equations, we get that

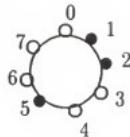
$$d(m, a) - d(x, y) = d(m, y) - d(x, a)$$

Since  $c_x < c_m$ , we have that  $d(m, y) - d(x, a) > 0$ . This implies that  $d(m, y) > d(x, a)$ . Similarly, we can show that  $d(m, z) \geq d(x, b)$ . Combining these two inequalities, we get that the path from  $y$  to  $z$  that goes through  $m$  is greater than the path from  $a$  to  $b$  that goes through  $x$ . But this contradicts the maximality of  $P$ , since by construction the first path does not include any nodes in  $D$ . Thus the lemma has to be true.  $\square$

The complement of path  $P$ , and thus the optimal root, can be found with an  $O(N)$  algorithm.

*Example 5.1 :*

Let  $N = 8$ . Let  $D = 1, 2, 5$  (shown in dark in the figure).



$P$  is  $(1, 0, 7, 6, 5)$ , and  $\overline{P} = (1, 2, 3, 4, 5)$ . The midpoint of  $\overline{P}$  is node 3, and this is the optimal root with  $c_3 = 2$ .  $\square$

## 5.2 Multiple Groups

Let  $n$  be the maximum distance possible from any root to any node in a shortest-path tree. In a ring,  $n = \lfloor N/2 \rfloor$ . Any multicast tree must have the same shape (see figure below). For a given root  $r$ , and a shortest-path tree rooted at  $r$ , let  $L_{ir}$  be the maximum distance from  $r$  to the last node of group  $i$  in the left side of the tree, and  $R_{ir}$  the corresponding distance in the right side, as shown in figure 5.1.

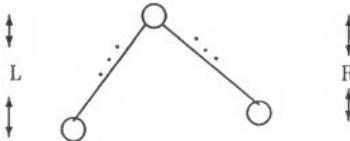


Figure 5.1. Form of tree in a ring network

Clearly,  $0 \leq L_{ir}, R_{ir} \leq n$  for all  $i$ . Before presenting the algorithm, let us see first how and why it works, as we describe the purpose of the variables used in it.

The algorithm first computes (going through the elements in  $D$ ) the values of  $R_{i0}$  and  $L_{i0}$  for all the groups. In general, once  $R_{ij}$  and  $L_{ij}$  are computed, to compute the values of  $R_{ij+1}$  and  $L_{ij+1}$ , since we are moving clockwise through the ring, we may need to subtract 1 from  $R_{ij}$  and add 1 to  $L_{ij}$ . The node farthest from node  $j$  to the right is now one unit closer to node  $j + 1$ , and the node farthest from  $j$  to the left is now one unit farther from  $j + 1$ . There are some special cases, however, that complicate things a little bit. The highest value that  $L_{ij}$  can get is  $n$ , so when  $L_{ij} = n$  we cannot increase it to compute  $L_{ij+1}$ . Also, the lowest value  $R_{ij}$  can get is 0, so again we need to compute  $R_{ij+1}$  in some other way when  $R_{ij} = 0$ . It is necessary to deal with these cases separately. There is one more special case. This one occurs when  $L_{ij} = 0$ .

When  $L_{ij} = 0$ , and we move to node  $j + 1$ ,  $L_{ij+1}$  does not necessarily get increased, but instead keeps its value of 0. Consider, for example, the ring with 5 nodes. If  $D = \{1, 2\}$ ,  $L_{10} = 0$ ,  $R_{10} = 2$ , but  $L_{11}$  remains at 0, while  $R_{11}$  decreases to 1. This situation occurs whenever all the nodes in  $D$  are covered by the right portion of the

tree, and  $j \notin D$ . The way to solve this case is to keep a counter that stores the distance between node  $j$  and the closest element to  $j$  in  $D$ . Essentially, the counter keeps track of how many iterations (when  $L = 0$ ) we would have to keep  $L = 0$ . This counter is called  $counter\_L0[i]$  in the algorithm, and it gets initialized every time  $L_{ij}$  becomes 0, with the distance between node  $j + 1$  and the node following  $j + 1$  in  $D$ .

When  $R_{ij} = 0$ , it means that the destination list is covered completely by the left side of the multicast tree when  $j$  is the root. It will remain at 0 until  $L$  becomes  $n$ .

Now, assume that  $L_{ij} = n$ . When  $j + 1$  is the root, then the node farthest from node  $j$  *counterclockwise* (let us call this node  $\bar{j}$ ) would be covered by the right side of the tree. If  $N$  is odd, the distance from  $j + 1$  to  $\bar{j}$  is  $n$ , while when  $N$  is even it is  $n - 1$ . This value is assigned to  $R_{i,j+1}$ . Now,  $L_{i,j+1}$  would be the distance between node  $j + 1$  and the node following  $\bar{j}$  in  $D$ ,  $\bar{j} + 1$ , if this node is not covered by the right side of the tree, which is the path  $(j + 1, j + 2, \dots, \bar{j})$ . Otherwise, the destination list is covered completely by this path, and  $L_{i,j+1}$  becomes 0. The value of  $\bar{j}$  is computed by  $\bar{j} = (N + j - n) \bmod N$ . Let  $CW_i(a)$  be the node in group  $i$  closest to node  $a$  clockwise, including  $a$  itself.

#### Algorithm 5.1 Ring-Multiple Groups

```

begin
1. for i=1 to p do
    compute  $R_{i0}, L_{i0}$ ;
    if  $L_{i0} = 0$  then  $counter\_L0[i] = CW_i(0)$ 
    else  $counter\_L0[i] = 0$ ;
2. for i=1 to p do
    for i=0 to N-2 do
        if  $L_{ij} = n$  then
            if N is even then  $R_{i,j+1} = n - 1$ 
            else  $R_{i,j+1} = n$ ;
```

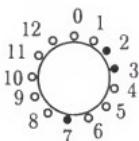
```

 $x = CW_i(\vec{j} + 1);$ 
if  $x \notin$  path  $(j+1, j+2, \dots, \vec{j})$ 
then  $L_{ij+1} = d(j+1, x)$ 
else
     $L_{ij+1} = 0;$ 
     $counter\_L0[i] = d(j+1, CW_i(j+1))$ 
else
    if  $counter\_L0[i] > 0$  then
         $R_{ij+1} = R_j - 1;$ 
         $L_{ij+1} = 0;$ 
         $counter\_L0[i] = counter\_L0[i] - 1;$ 
    else
        if  $R_{ij} \neq 0$  then  $R_{ij+1} = R_{ij} - 1$  else  $R_{ij+1} = 0;$ 
         $L_{ij+1} = L_{ij} + 1;$ 
3. select an optimal root  $j$  that minimizes
 $\sum_{i=1}^p \max\{L_{ij}, R_{ij}\}$ 
end;

```

Example 5.2 :

Consider the following ring, with  $D = 2, 3, 7$ .



The algorithm computes the following values for  $R$  and  $L$ :

Node j	$R_j$	$L_j$
0	3	6
1	6	0
2	5	0
3	4	1
4	3	2
5	2	3
6	1	4
7	0	5
8	0	6
9	6	6
10	6	3
11	5	4
12	4	5

Nodes 4 and 5 are both optimal with cost 3.  $\square$

The algorithm requires then  $O(Np)$  time complexity. This is better than the  $O(Nm)$  required to compute the cost of every node going through all the elements of the destination lists for every node. Note that  $m$  itself may be in the order of  $Np$ .

### 5.3 Primary Destinations

The primary destinations for the multicast tree constructed for rings can be computed again in  $O(p)$ . Let  $L_{ij}$  and  $R_{ij}$  be the values computed by Algorithm 5.1 for group  $i$  and node  $j$ . The optimal root found by the algorithm is the node  $j$  that minimizes  $\sum_{i=1}^p \max\{L_{ij}, R_{ij}\}$ . For each group  $i$ , the primary destination  $PD_i$  is computed as follows:

```

if ( $L_{ij} > 0$ ) and ( $R_{ij} > 0$ ) then  $PD_i = root$ 
else if  $L_{ij} = 0$  then  $PD_i = CW_i(root)$ 
else  $PD_i = CCW_i(root)$ 

```

where  $CCW_i(root)$  is the closest node to the root counterclockwise.

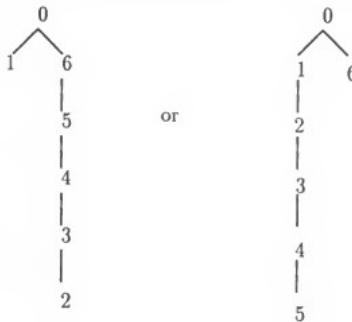
### 5.4 Minimizing the Real Cost of the OMT

For any shortest-path tree the sum of the distances from the primary destinations to the nodes farther from them in each group can be minimized in the following way. Instead of finding the value that minimizes  $\sum_{i=1}^p \max\{L_{ij}, R_{ij}\}$ , we find the node  $r$  that minimizes the following expression:

$$\sum_{i=1}^p \begin{cases} \max\{L_{ij}, R_{ij}\} & \text{if } L_{ij}, R_{ij} > 0 \\ R_{ij} - d(r, x) & \text{if } L_{ij} = 0 \ (x = CW_i(r)) \\ L_{ij} - d(r, x) & \text{if } R_{ij} = 0 \ (x = CCW_i(r)) \end{cases}$$

From all the shortest path trees, this method will give us the optimal root in terms of the distances from the primary destinations. However, using this method we always produce a shortest path tree, and the optimal multicast tree does not

necessarily have this property. Consider, for example, the ring with  $N = 7$ , and groups 016, 23, 34, and 45. The OMT in this case is given by:



None of these trees is a shortest path tree, and thus the method given above will not construct the OMT. Let  $R_{ijk}$  and  $L_{ijk}$  be the cost of the right and left side of the multicast tree for group  $i$  when node  $j$  is the root, and exactly  $k$  elements of group  $i$  are covered by the right side of the tree. Note that now the tree does not have to be a shortest-path tree. Let  $d^{ccw}(x, y)$  and  $d^{cw}(x, y)$  be the distances between nodes  $x$  and  $y$  counterclockwise and clockwise, respectively. Also, let  $CW_i(a)$  (respectively  $CCW_i(a)$ ) be the node in group  $i$  closest to node  $a$  clockwise, including node  $a$ . The following algorithm, which is  $O(Nmp)$ , constructs the OMT for any ring network, and finds the primary destinations. Remember that  $m = \sum_{i=1}^p m_i$ , and  $m_i$  is the number of nodes in group  $i$ .

The values of  $R_{ijk}$  and  $L_{ijk}$  are computed in the following way. When  $k = 0$ , all nodes in group  $i$  must be covered by the left side of the tree, so  $R_{ijk} = 0$ , and  $L_{ijk}$  is the counterclockwise distance between node  $j$  and the closest element in the group to node  $j + 1$  clockwise; for other values of  $k$ ,  $R_{ijk}$  is the clockwise distance

between  $j$  and  $x$ , where  $x$  is the closest node clockwise to the one used in the previous iteration when  $k$  was  $k - 1$ , while  $L_{ijk}$  has to cover all nodes following node  $x$  in the group. When  $k = m_i$ , then all nodes are covered by the right side of the tree, and thus  $L_{ijk} = 0$ . If node  $j$  is an element of group  $i$ , then only  $m_i - 1$  nodes have to be covered by the tree, and  $x$  will be  $j$  again when  $k = m_i - 1$ . In this case we set  $L_{ijm_i}$  and  $R_{ijm_i}$  to their previous values and  $k$  is increased to exit the loop.

*Algorithm 5.2 OMT for ring networks*

```

begin
  for i=1 to p do
    for j=0 to N-1 do
      x=j;
      for k=0 to  $m_i$  do
         $R_{ijk} = d^{cw}(j, x);$ 
        if  $k = m_i$  then  $L_{ijk} = 0$ 
        else  $L_{ijk} = d^{wc}(j, CW_i(x + 1));$ 
         $x = CW_i(x + 1);$ 
        if  $x = j$  then
           $L_{ijk+1} = L_{ijk};$ 
           $R_{ijk+1} = R_{ijk};$ 
           $k=k+1;$ 
         $COST_{ijk} = \begin{cases} \max\{L_{ijk}, R_{ijk}\} & \text{if } L_{ijk}, R_{ijk} > 0 \\ R_{ijk} - d(j, CW_i(j)) & \text{if } L_{ijk} = 0 \\ L_{ijk} - d(j, CCW_i(j)) & \text{if } R_{ijk} = 0 \end{cases}$ 
      end;
    end;
  end;

```

The *OMT* can be found minimizing  $COST_{jk} = \sum_{i=1}^p COST_{ijk}$ . The optimal root and the routing are given by the indexes  $j$  and  $k$  that minimize  $COST_{jk}$ .

### 5.5 Multiple Trees

The three steps as discussed in section 4.5 to build the multicast tree  $T_i$  for group  $i$  once the subtrees for the connected components of the graph  $G$  as defined have been built can be used, as mentioned before, by the different network topologies we

consider. We need to discuss how to perform the third step of the algorithm, that is, how to connect the nodes in set  $A$ .

First we compute the optimal root  $r'$  for the set  $A$  assuming that all nodes in  $A$  have depth 0. To do this the algorithm described in section 5.1 to compute the optimal root for a single group can be used. Now we determine the nodes  $a, b \in A$  with the following properties:

- $distance(r', a) = \text{clockwise distance } (r', a)$
- $distance(r', a) + depth(a)$  is maximum for all the nodes in  $A$  satisfying the previous property
- $distance(r', b) = \text{counterclockwise distance } (r', b)$
- $distance(r', b) + depth(b)$  is maximum for all the nodes in  $A$  satisfying the previous property

Taking  $r'$  as the root, let  $x$  and  $y$  be the costs of the left (clockwise) and right (counterclockwise) sides of the multicast tree. The cost of the root node  $r'$  is thus  $\max\{distance(r', a) + cost(a), distance(r', b) + depth(b)\} = \max\{x, y\}$ . Essentially, to compute the optimal root, we need to move  $|x - y| / 2$  nodes in the direction of node  $a$  if  $x \geq y$  or in the direction of  $b$  if  $y > x$ . Then we would obtain the midpoint  $r$  between  $distance(r', a) + depth(a)$  and  $distance(r', b) + depth(b)$ . The cost of the left side of the tree would be as close as possible to the cost of the right side. Clearly, this would yield the optimal root.

The final tree has the same form as the tree for linear networks shown in figure 4.1.

Simulations ran for the ring network with 64 nodes under the same conditions as the ones ran for linear networks show essentially the same results. The cost of the multicasts is better for the multiple trees approach, while at the same time it generates more traffic. The primary destinations, as for linear networks, have much more variability under the multiple trees approach.

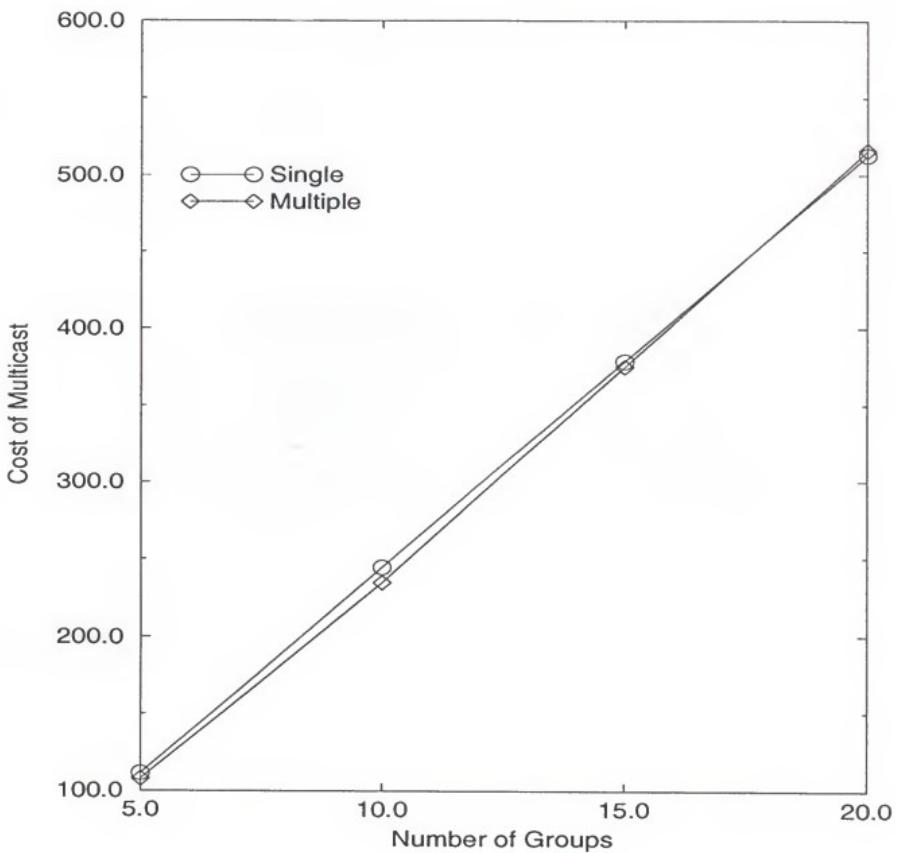


Figure 5.2. Cost for Ring Network ( $N=64$ )

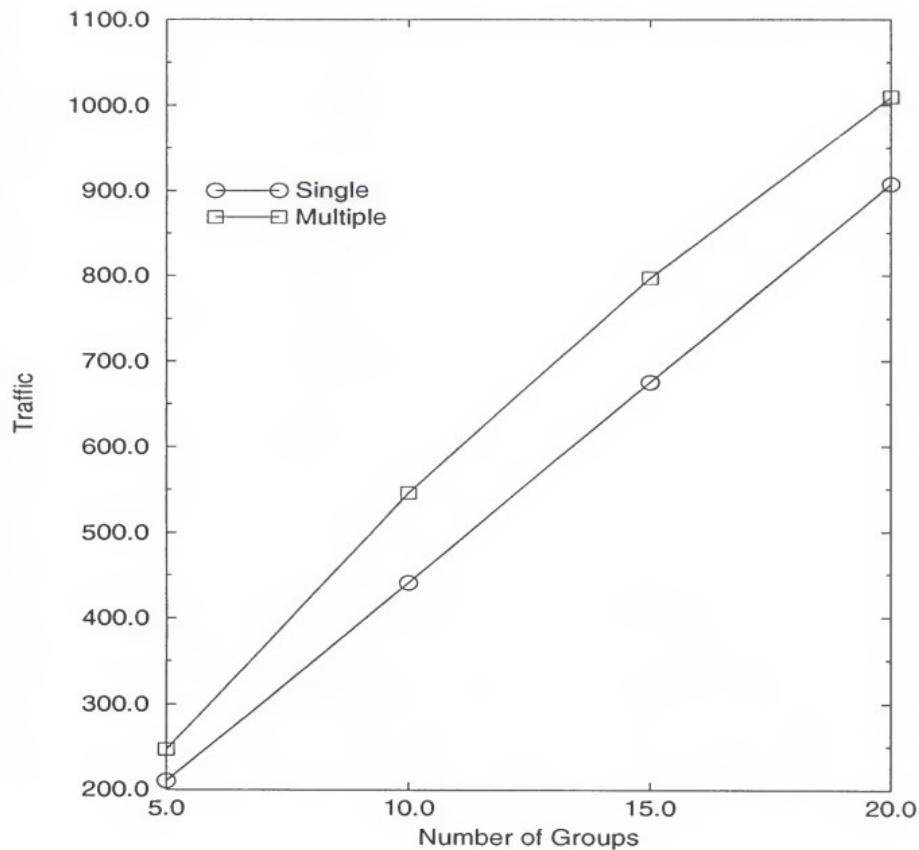


Figure 5.3. Traffic for Ring Netwrok (N=64)

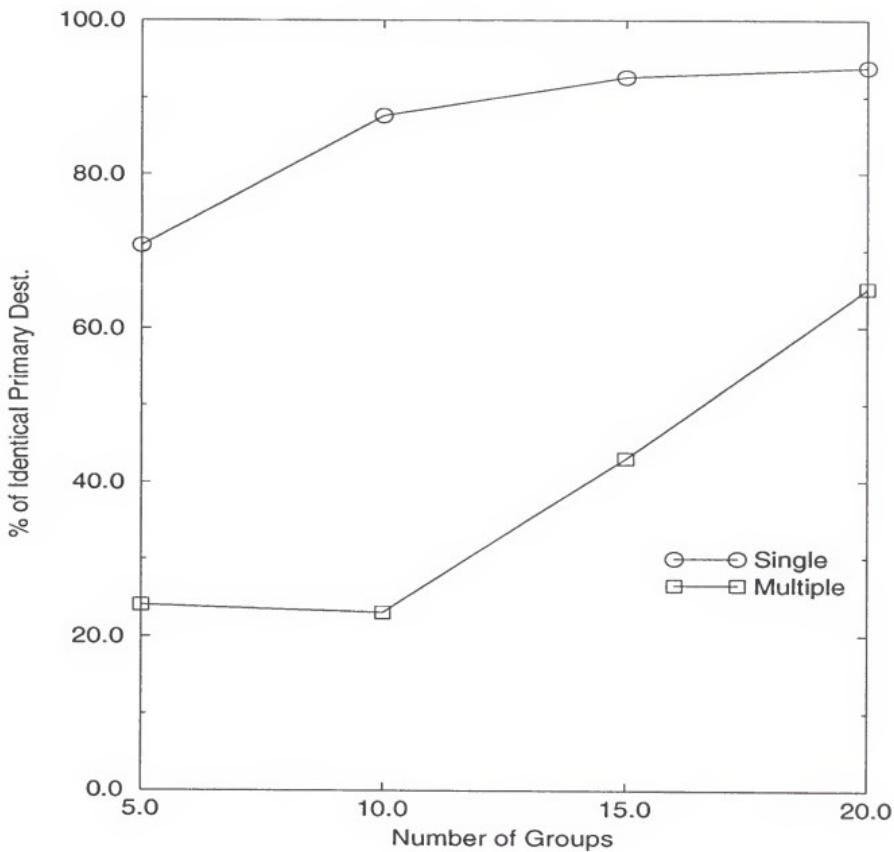


Figure 5.4. Primary Destinations for Ring Netwrok (N=64)

## CHAPTER 6

### MULTICAST TREES FOR MESH NETWORKS

#### 6.1 Single Group

Let  $M$  be an  $m \times n$  mesh. Let  $D$  be a set of points from the grid. The cost of node  $i$  is, as before, defined by  $c_i = \max_{j \in D} d(i, j)$ . The distance in a grid between the points  $(x_1, x_2)$  and  $(y_1, y_2)$  is given by  $|x_1 - y_1| + |x_2 - y_2|$ . The first thing we need to do is find the node  $r$  with minimum cost.

*Lemma 6.1* *Let  $y$  be an optimal solution. For any node  $x$ , if  $c_y < c_x$ , then  $x$  has a neighbor or a diagonal  $z$  such that  $c_z < c_x$ .*

**Proof:**

Let  $x = (i, j)$ ,  $y = (k, l)$ . Assume that  $c_y < c_x$ . This implies that  $x \neq y$ . Assume, w.l.g., that  $i < k$ . Let  $\max_{n \in D} d(a, n) = d(a, m_a)$ , For any node  $a$ ,  $m_a$  is the node in  $D$  farthest from node  $a$ .<sup>1</sup> Since  $y$  is optimal, we know that  $d(y, m_y) < d(x, m_x)$ , and  $d(y, m_y) \leq d(z, m_z)$  for any neighbor  $z$ .

*Case 1:*  $x$  and  $y$  have the same second component ( $j = l$ ). Then choose  $z = (i+1, j)$ . We need to show that  $c_z < c_x$ . The costs of  $z$  and  $x$  are given by  $c_z = d(z, m_z)$  and  $c_x = d(x, m_x)$ . We know that  $y$  is at least as close to  $m_z$  as  $z$  is

---

<sup>1</sup>Note that  $m_a$  may not be unique.

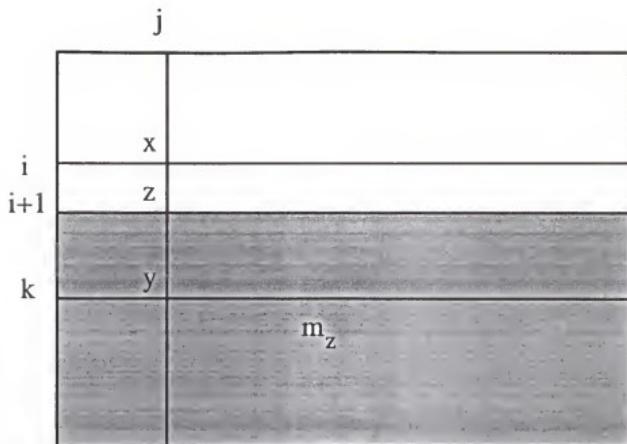


Figure 6.1.

(note that maybe  $y = z$ ). Thus, the first component of  $m_z$  is at least  $i + 1$ . Given  $x$  and  $y$  as shown in figure 6.1,  $m_z$  must be somewhere in the shaded area.

Since  $x$  and  $z$  have the same second component, it is clear that  $d(z, m_z) < d(x, m_z) \leq d(x, m_x)$ . Thus,  $c_z < c_x$ .

*Case 2:*  $x$  and  $y$  have different second component. Assume, *w.l.g.*, that  $l > j$ . That is,  $y$  is located below and to the right of  $x$  in the grid. Let  $z_1 = (i + 1, j), z_2 = (i, j + 1), z_3 = (i + 1, j + 1)$ . We need to show that  $c_{z_i} < c_x$  for  $i=1,2$ , or 3. We prove this result by contradiction.

Assume that  $d(x, m_x) \leq d(z_i, m_{z_i}) \forall i = 1, 2, 3$ . We know that  $d(x, m_{z_1}) \leq d(x, m_x) \leq d(z_1, m_{z_1})$ . That is,  $x$  is closer to  $m_{z_1}$  than  $z_1$  is. This implies that  $m_{z_1}$  has first component  $\leq i$ . A similar relation for  $m_{z_2}$  shows that it has second component  $\leq j$ . The fact that  $d(y, m_{z_i}) \leq d(z_i, m_{z_i})$  for  $i=1,2$  implies that  $m_{z_1}$  has second component

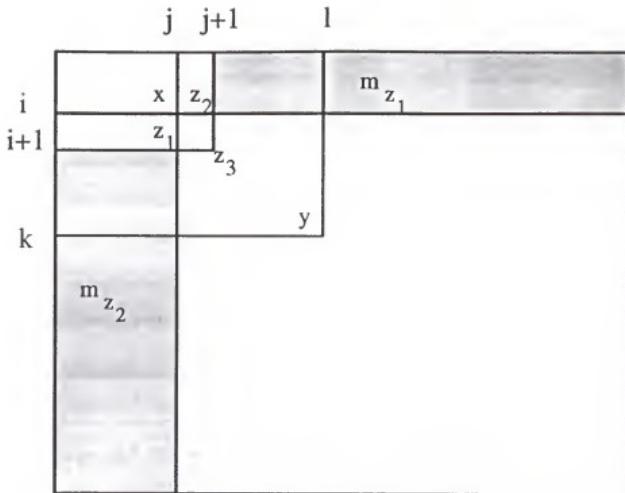


Figure 6.2.

$\geq j + 1$ , and that  $m_{z_2}$  has first component  $\geq i + 1$ . Also,  $d(x, m_x) \leq d(z_3, m_{z_3})$ , which implies that  $m_x$  has first component  $\leq i$  or second component  $\leq j$ , but not both (since  $d(y, m_x) < d(x, m_x)$ ). Given  $x$  and  $y$  as shown in figure 6.2,  $m_{z_1}$  and  $m_{z_2}$  must be in the shaded areas, while  $m_x$  can be in either of them.

Now, consider a shortest path from  $m_{z_1}$  to  $x$ , and from  $x$  to  $m_{z_2}$ . This path has to be a shortest path from  $m_{z_1}$  to  $m_{z_2}$ .  $x$  must be the midpoint of this path. That is,  $d(m_{z_1}, m_{z_2}) = d(m_{z_1}, x) + d(x, m_{z_2})$ , and  $|d(x, m_{z_1}) - d(x, m_{z_2})| \leq 1$ . This last inequality must hold, or otherwise, if for example  $d(x, m_{z_1}) > d(x, m_{z_2}) + 1$ , then  $m_{z_1}$  would be the point farther from  $z_2$ , and not  $m_{z_2}$  as it is.

Since  $x$  and  $z_i$  are adjacent, and  $d(x, m_x) \leq d(z_i, m_{z_i})$  for  $i=1,2$ , this implies that  $d(x, m_x) < d(z_i, m_{z_i})$  for  $i=1,2$ .<sup>2</sup> Thus it must be the case that  $d(x, m_x) = \max(d(x, m_{z_1}), d(x, m_{z_2}))$ . Assume, *w.l.g.*, that  $d(x, m_{z_1}) \geq d(x, m_{z_2})$ .  $d(y, m_x) < d(x, m_x)$  implies that  $d(y, m_{z_1}) < d(x, m_{z_1})$ . Therefore,  $d(y, m_{z_2}) > d(x, m_{z_2}) \geq d(x, m_x) - 1$ .<sup>3</sup> Thus,  $d(y, m_{z_2}) \geq d(x, m_x)$ . This implies that  $c_y \geq c_x$ , which is a contradiction. Therefore, the statement of the lemma must be true.  $\square$

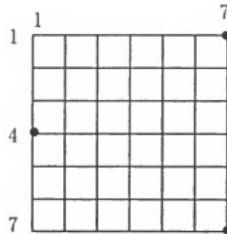
The lemma states that if a node is not an optimal root, then it must have a neighbor or a diagonal with lower cost. The idea then is to start with a good candidate for an optimal solution, and then if it turns to be not an optimal solution, we move in the direction of an optimal until we eventually converge to one. We select as the initial choice the node such that the first and second components are the midpoints of the lists of first and second components, respectively, of the elements in the multicast group. Then we check if that node has a neighbor with less cost. If it doesn't, then it is optimal. Otherwise we move to that new node, and repeat, until we reach an optimal solution.

In a linear network, computing the midpoint of the smallest node and the highest node gives us the optimal root. In a 2-dimensional mesh, computing the midpoint that way for each dimension separately to get the two coordinates of a point not necessarily will produce the optimal root. The distance from that point to the lowest and highest rows and columns in the mesh containing elements of  $D$  would be minimized, but the *sum* of the horizontal and vertical distances to reach the farthest *node* may not

<sup>2</sup>Two adjacent nodes cannot have the same distance to any node.

<sup>3</sup>The first inequality follows from the fact that the path from  $m_{z_1}$  to  $z$ , and from  $z$  to  $m_{z_2}$  is a shortest path from  $m_{z_1}$  to  $m_{z_2}$  together with the fact that  $d(y, m_{z_1}) < d(x, m_{z_1})$ .

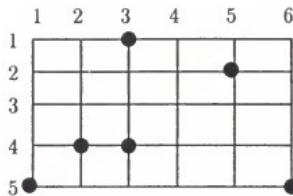
be minimized. Let  $h$  be the maximum possible of nodes to which we would have to jump in the algorithm to compute the optimal root. To find the value of  $h$ , a worst case scenario should be considered. An example of this is illustrated in the figure below.



Let  $D$  be as shown in the figure. The first choice for the root is the node  $(4, 4)$ . This point minimizes the distance to the lowest column and row with elements from  $D$ . However, it is not optimal. Let  $d_1$  be the distance from that point to the lowest (or highest) column that contains an element of  $D$ , and  $d_2$  the corresponding distance to the lowest (or highest) row. The maximum cost of the point computed is  $d_1 + d_2$ , as it is in our example. From one side of the potential tree the cost is  $d_1 + d_2$ , and from the other side it is only  $d_1$  (in this case). We would like to "balance" these values. Moving  $h$  nodes to the right, we can balance these distances, making  $d_1 + h$  as close as possible to  $d_1 + d_2 - h$ . This implies that  $h = \lfloor d_2/2 \rfloor$ .  $d_2$  is at most  $\lfloor m/2 \rfloor$ , so  $h$  is  $O(m/4)$ . In this example,  $d_1 = d_2 = 3$ , and  $h = 1$ . The algorithm thus has time complexity  $O(m | D |)$  in the worst case.

Example 6.1 :

Let  $M$  be the  $5 \times 6$  grid, and let  $D = \{(1, 3), (2, 5), (4, 2), (4, 3), (5, 1), (5, 6)\}$ , as shown in the figure below.



Our first choice for optimal root is the node  $x = (\lfloor (1+5)/2 \rfloor, \lfloor (1+6)/2 \rfloor) = (3, 3)$ .  $c_{3,3} = 5$ . Now we check if any neighbor or diagonal has a lower cost. In fact, node  $(4,3)$  has cost 4. This node does not have a neighbor or a diagonal with lower cost, hence it is optimal.  $\square$

The following lemma will be useful for the solution of the multiple group case.

Lemma 6.2 Let  $x=(i,j)$ ,  $y=(k,l)$ ,  $k \geq i$ ,  $l > j$ ,  $c_y < c_x$ , and  $z_3 = (i + 1, j + 1)$ . Then  $c_{z_3} \leq c_x$ .

**Proof:**

The only way that  $d(x, m_x)$  could be smaller than  $d(z_3, m_{z_3})$  is if  $m_x$  has first component  $\leq i$ , and second component  $\leq j$ . This cannot be the case, since  $d(y, m_x) < d(x, m_x)$ . Thus the lemma is true.  $\square$

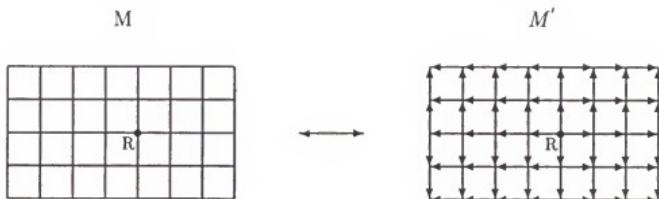
The problem now is to determine the routing to construct the multicast tree using the optimal root.

The problem of finding a Steiner tree given a set of points in a plane using rectilinear distances has been proved to be an NP-Complete problem [15]. Here we propose

a heuristic algorithm to find a suboptimal multicast tree (SOMT). We wish to find a tree with the following properties:

- $V(T) \subseteq V(M)$
- the maximum of the distances from the root to every node in the destination list is as small as possible
- if  $R$  is the root of the tree, then the following holds:  $d_T(x, R) = d_M(x, R)$   
 $\forall x \in SOMT$
- no tree exists satisfying the three properties above that has fewer nodes than this one

The third condition above implies that the heuristic algorithm will construct a shortest-path tree for the nodes in the multicast group  $D$ . Constructing a shortest-path tree with lowest traffic rooted at node  $R$  (grid  $M$  in the figure below) is equivalent to finding a rectilinear directed Steiner tree rooted at  $R$  in a directed mesh in which all the links are directed away from  $R$  ( $M'$  in the figure).



The problem of finding a rectilinear directed Steiner tree has been known in the literature as the Rectilinear Steiner Arborescence (*RSA*) problem. The problem here is to find a Steiner directed tree for a set  $D$  of nodes lying in the grid, rooted at point

$R$ , using only horizontal and vertical arcs oriented away from  $R$ . Rao et al. [32] have studied the Rectilinear Steiner Arborescence problem when the set  $D$  lies in the first quadrant of  $E^2$ , and the directed tree is rooted at the origin. They proposed a heuristic algorithm in which a Steiner tree  $H$  for  $D$  is generated by iteratively substituting  $\min(p, q)$  for pairs of points  $p, q$  in  $D$  until the origin remains, where  $p = (x_p, y_p)$ ,  $q = (x_q, y_q)$ , and  $\min(p, q) = (\min(x_p, y_p), \min(x_q, y_q))$ . The points  $p$  and  $q$  are chosen to maximize  $\| \min(p, q) \|$  over all the points currently in  $D$ . They have shown that  $l(H) \leq 2l(RSMA)$ , where  $l(A)$  is the number of links in the tree  $A$  and  $RSMA$  is an RSA  $A$  with minimum  $l(A)$ .

A similar heuristic is presented here that generates a *SOMT* for an  $m \times n$  grid when  $D$  lies anywhere in the grid. For two grid points  $p, q$ , let  $\max_o(p, q)$  be the grid point that maximizes the overlapping in the shortest paths from the origin to the points  $p$  and  $q$ .  $\max_o(p, q)$  is defined in the following way, given that  $p = (x_p, y_p)$ ,  $q = (x_q, y_q)$ , for  $z = x$  or  $y$  coordinates: if  $z_p, z_q \geq 0$  then the  $z$ -coordinate of  $\max_o$  is given by  $\min(z_p, z_q)$ . Else if  $z_p, z_q \leq 0$  then the  $z$ -coordinate is  $\max(z_p, z_q)$ . Otherwise it is 0. We assume that the root of the tree is the origin.

Let  $L_z$  denote the four lines lying in the  $m \times n$  grid given by  $|x| + |y| = z$ . A set of points  $P$  is a *cover* for a set of points  $Q$  if  $\forall q \in Q$  there exists a point  $p \in P$  with a path from  $p$  to  $q$ . Let  $Q_z$  be the subset of  $Q$  lying above  $L_z$ . Then let  $MC(Q, z)$  denote the set of points on  $L_z$  with minimum cardinality that covers  $Q_z$ . It follows directly from this definition that any RSA for  $D$  intersects  $L_z$  at least  $MC(D, z)$  times for every  $z$ . The nodes in the initial set  $D$  are called *sinks*.

A heuristic algorithm generates a *SOMT* by iteratively substituting two points in  $D$  by the point  $\max_o(p, q)$  until the origin remains. Again, the pair of points  $p, q$  are chosen to maximize  $\| \max_o(p, q) \|$ .

Let  $D_z$  denote the set  $D$  at the point in the construction of the *SOMT* in which the last possible pair of points  $p, q$  having  $\| \max_o(p, q) \| \geq z$  have been joined, and let  $W_z = \{p \in D_z : \| p \| \geq z\}$ .

*Lemma 6.3* During the construction of the *SOMT*, for any  $z \geq 0$ , there can be at most one point in  $W_z$  on or above the horizontal line  $y = (0, z)$  (also, there is at most one point in  $W_z$  on or below the horizontal line  $y = (0, -z)$ , at most one point on or to the right of the vertical line  $x = (z, 0)$  and at most one on or to the left of the vertical line  $x = (-z, 0)$ ).

#### Proof:

Let  $p, q \in W_z$  with  $y_p, y_q \geq z$ . Let  $y_p \geq y_q$ . Then  $\max_o(p, q) = (x, y_q)$  for some value  $x$ . Hence  $\| (x, y_q) \| \geq z$ , contradicting the fact that  $p, q \in W_z$ . A similar proof works for the other 3 lines.  $\square$

The following lemma is an adapted version of lemma 4 in [32].

*Lemma 6.4* Let  $p_i = (x_i, y_i)$  be a point in a *SOMT* generated by the heuristic. If  $x_i \neq 0$  and  $y_i \neq 0$ , then there must exist a sink on the vertical path  $x = x_i$  (horizontal path  $y = y_i$ ) starting at  $p_i$ .

#### Proof:

Assume, *w.l.g.*, that  $p_i$  lies in the first quadrant. The proof is by induction on the

number of descendants of  $p_i$ . If  $p_i$  does not have descendants then it is a sink, and the result immediately follows. Suppose, on the contrary, that  $p_i = \max_o(p_j, p_k)$ . Observe that these two points belong also to the first quadrant. Thus  $x_i = \min(x_j, x_k)$ , say  $x_k$ , and  $y_k \geq y_i$ . By induction there is a sink on the vertical path  $x = x_k$  starting at  $p_j$ . The fact that  $y_k \geq y_i$  implies that this sink is also on the vertical path  $x = x_i = x_k$  staring at  $p_i$ . A similar argument can be made for the  $y$ -coordinate.  $\square$

Now assume that the points  $p$  and  $q$  lie in the first quadrant. The following two lemmas are taken directly from [32]. The proofs can be found in their paper.

*Lemma 6.5* *During the construction of the SOMT, for any  $z \geq 0$  and  $p, q \in W_z$ ,*

1.  $x_p \neq x_q$  and  $y_p \neq y_q$ ,
2.  $x_p < x_q$  if and only if  $y_p > y_q$ ,
3. if  $x_p < x_q$ , then  $x_p + y_q < z$ .

*Lemma 6.6* *Let  $p, q \in W_z$  with  $x_p < x_q$ . Then there exists a sink either on the horizontal path between  $p$  and  $(x_p, y_p)$  or on the vertical path between  $(x_q, y_p)$  and  $q$ .*

It is straight forward to prove similar results when both of the points  $p$  and  $q$  lie in any of the other three quadrants. The following theorem provides an upper bound for the traffic generated for the multicast tree by the heuristic. Its proof is similar to the proof of the upper bound provided in [32].

*Theorem 6.1*  $l(\text{SOMT}) \leq 2l(\text{RSMA})$ .

**Proof:**

First, note that for any RSA  $A$ ,  $l(A) = \sum_z (A \cap L_z)$ . We know that  $|L_z \cap A| \geq |MC(D, z)|$  for any RSA  $A$ . Thus it is sufficient to show that  $|SOMT \cap L_z| \leq 2 |MC(D, z)|$  for any  $z$ .

For any given  $z$ , order the points in  $W_z$  into  $p_1, p_2, \dots, p_{m(z)}$  circularly as follows: first take the nodes in the first quadrant and order them by increasing values of the  $x$ -coordinates (with  $x_1 > 0$ ), then the nodes in the second quadrant by decreasing order of the  $x$ -coordinate, and so on.

It can be shown that  $MC(D, z)$  must contain distinct points for each pair of points  $p_{2i-1}, p_{2i}$ , for  $i = 1, 2, \dots, \lfloor m(z)/2 \rfloor$ . Consider the two pairs of points  $(p_{2i-1}, p_{2i})$ , and  $(p_{2i+1}, p_{2i+2})$ , for any given  $i$  in the above range. Assume, for simplicity, that these points lie in the first quadrant. By Lemma 6.6 there must exist a sink on the horizontal path  $y = y_{2i-1}$  between  $p_{2i-1}$  and  $(x_{2i}, y_{2i-1})$  or on the vertical path  $x = x_{2i}$  between  $(x_{2i}, y_{2i-1})$  and  $p_{2i}$ . Hence  $MC(D, z)$  must contain a point whose  $x$ -coordinate lies between  $z - y_{2i-1}$  and  $x_{2i}$ . Also, there is a sink on the horizontal path  $y = y_{2i+1}$  between  $p_{2i+1}$  and  $(x_{2i+2}, y_{2i+1})$  or on the vertical path  $x = x_{2i+2}$  between  $(x_{2i+2}, y_{2i+1})$  and  $p_{2i+2}$ . Hence  $MC(D, z)$  must contain a point whose  $x$ -coordinate lies between  $z - y_{2i+1}$  and  $x_{2i+2}$ . Since  $x_{2i} + y_{2i+1} < z$  by Lemma 6.5, then the above two points are different. A similar result holds no matter where the points lie.

Thus  $W_z$  contains at most  $2 \lfloor |MC(D, z)| \rfloor$  points. This implies that if  $m(z)$  is even, then the  $SOMT$  generated by the heuristic intersects  $L_z$  at most  $2 |MC(D, z)|$  times. If  $m(z)$  is odd, then consider  $p_1$ .

*Case 1:*  $x_{p_1} \neq 0$ .

By Lemma 6.4, there is a sink  $s$  on the vertical path  $x = x_{p_1}$  starting at  $p_1$ . Thus  $MC(D, z)$  must contain a point whose  $x$ -coordinate is between 0 and  $x_{p_1}$ . For the rest of the  $|W_z| - 1$  points in  $W_z$  there correspond at least  $(|W_z| - 1)/2$  distinct points in  $MC(D, z)$ , and it is easy to see that these points must be different from the point in  $MC(D, z)$  corresponding to  $p_1$ .<sup>4</sup> Thus  $MC(D, z)$  contains at least  $|W_z|/2$  distinct points.

*Case 2:*  $x_{p_1} = 0$ . We can assume now that  $W_z$  contains points on or above the points  $(0, z), (z, 0), (0, -z), (-z, 0)$  in the same vertical or horizontal lines. Otherwise, we could order the points in  $W_z$  starting with the points in some other quadrant and reduce the problem to Case 1.

Assume, *w.l.g.*, that there are  $2c$  points in  $W_z$  in the first quadrant, for some positive integer  $c$ . Note that since now we are assuming that  $|W_z|$  is odd, at least one quadrant must have an even number of points in  $W_z$ . Thus  $MC(D, z)$  must contain at least  $c$  points in the first quadrant.

*Case 2a:*  $MC(D, z)$  contains at least  $c + 1$  points in the first quadrant.

$MC(D, z)$  must contain at least  $\lfloor (|W_z| - 2c)/2 \rfloor$  distinct points in the other three quadrants. Thus  $MC(D, z)$  contains at least  $\lfloor |W_z|/2 \rfloor - c + c + 1 > |W_z|/2$  distinct points.

*Case 2b:*  $MC(D, z)$  contains exactly  $c$  points in the first quadrant.

---

<sup>4</sup>We can assume here that  $\|p_{m(z)}\| < z$ . Otherwise, we can order the nodes circularly the other way around, starting with  $p_{m(z)}$  and finishing with  $p_1$ .

We will show that at least one of the points  $(0, z)$  or  $(z, 0)$  cannot be one of these  $c$  points in  $MC(D, z)$ . Consider the first  $2c - 2$  points in  $W_z$  in the first quadrant, not including  $p_1$ . There must be  $c - 1$  distinct points in  $MC(D, z)$  to cover them, and none of them is  $(0, z)$  or  $(z, 0)$ . Thus one of these two points is not in  $MC(D, z)$ . Assume, *w.l.g.*, that  $p_1 \notin MC(D, z)$ . Thus between  $x_{p_{m(z)}}$  and 0 there is a different point in  $MC(D, z)$  required to cover the sink on or above  $p_{m(z)}$ . For the rest of the  $|W_z| - 2c + 1$  points in  $W_z$  there must be  $(|W_z| - 2c + 1)/2$  distinct points in  $MC(D, z)$ . Thus  $MC(D, z)$  contains at least  $\lfloor |W_z|/2 \rfloor - c + c + 1 > |W_z|/2$  distinct points.

Thus  $W_z$  contains at most  $2 |MC(D, z)|$  points, and therefore the *SOMT* generated intersects  $L_z$  at most  $2 |MC(D, z)|$  times. Since  $z$  is arbitrary this holds for any  $z$ , and the theorem is proved.  $\square$

The implementation presented in [32] for their algorithm can be adapted for our algorithm. The key idea is that Steiner points can only be introduced by the heuristic from adjacent points in  $W_z$ . To implement the algorithm a priority queue is used containing all the sinks below the lines in  $L_z$  and all the  $|W_z| - 1$  potential new Steiner points  $\max_o(p, q)$  for each pair of adjacent points  $p, q \in W_z$ . See [32] for details of the implementation.

To evaluate the performance of the algorithm we ran simulation programs for an  $8 \times 8$  mesh both for the optimal and the heuristic algorithm. The optimal solution for 100 randomly generated groups was found by exhaustive search of all shortest-path trees, and compared to the suboptimal solution found by the heuristic algorithm for

the same multicast groups. The traffic is the number of links in the multicast tree. The results are shown in the following graph. To get the worst case line we added to the optimal traffic for each number of destinations the highest difference found between the optimal traffic and the traffic computed by the heuristic. It can be seen that even in the worst case the values computed are much closer to the optimal values than to the upper bound.

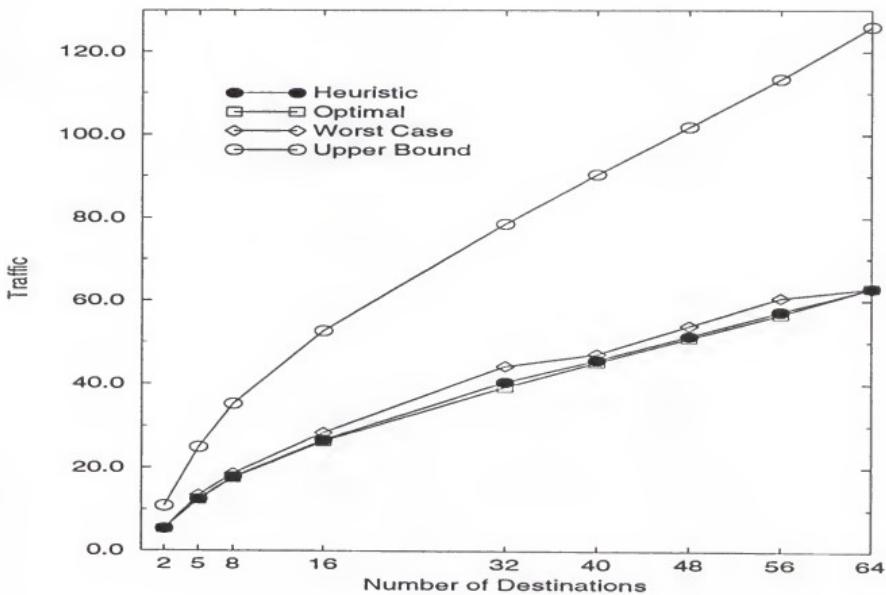


Figure 6.3. Optimal Algorithm vs Heuristic ( $N=64$ )

## 6.2 Multiple Groups

The solution of this problem is very similar to the solution provided for the single group case. We will give a generalization of the algorithm to find the optimal root given the  $m^*n$  mesh and the  $p$  groups. Once we have computed the optimal root, the heuristic algorithm proposed in the previous section can be used to construct the SOMT.

Now we want to find the node  $i$  such that

$$c_i = \sum_{j=1}^p \max_{1 \leq k \leq m_j} d(i, n_{jk})$$

is minimized, where  $n_{jk}$  is the  $k$ th element of the  $j$ th destination list, and  $m_j$  is the number of nodes in the  $j$ th group. Let  $\max_{1 \leq k \leq m_j} d(a, n_{jk}) = d(a, m_{ja})$ , for any node  $a$ .  $m_{ja}$  is now the node in group  $j$  farther away from node  $a$ .

Lemma 6.7 *Let  $y$  be an optimal solution. For any node  $x$ , if  $c_y < c_x$ , then  $x$  has a neighbor or a diagonal  $z$  such that  $c_z < c_x$ .*

**Proof:**

As before, let  $x = (i, j)$ ,  $y = (k, l)$ . Assume, *w.l.g.*, that  $x$  and  $y$  differ in their first component, with  $i < k$ , and that  $j \leq l$ . Again, define  $z_1 = (i+1, j)$ ,  $z_2 = (i, j+1)$ , and  $z_3 = (i+1, j+1)$ . Let  $c_{jn}$  be the cost of node  $n$  when only group  $j$  is considered (as a single group).

*Case 1:*  $c_{jz_3} < c_{jx}$  for some  $j$ ,  $1 \leq j \leq p$ . An immediate corollary of Lemma 6.2 is that  $c_{jz_3} \leq c_{jx} \forall 1 \leq j \leq p$ . Thus  $c_{z_3} < c_x$ , and the lemma is true.

*Case 2:*  $c_{jz_3} = c_{jx} \forall j, 1 \leq j \leq p$ . Thus,  $c_{jz_i} < c_{jx}$  for  $i=1$  or  $2$ . Note also that  $|c_{jz_i} - c_{jx}| \leq 1$ .

If  $p$  is odd, then  $c_{jz_i} < c_{jx}$  for at least  $\lceil p/2 \rceil$  groups for either  $i=1$  or  $i=2$ . Thus  $c_{z_i} < c_x$  for this value of  $i$ , and we are done. If  $p$  is even, we may think that maybe  $c_{jz_1} < c_{jx}$  for  $p/2$  groups, but for the other  $p/2$  groups,  $c_{jz_1} > c_{jx}$ , and that the same thing occurs with  $z_2$ , and thus none has a cost lower than  $x$ . We will show that this cannot happen. We just need to show that for 2 groups, the case in which  $c_{1z_1} < c_{1x}$ ,  $c_{1z_2} > c_{1x}$ , and  $c_{2z_1} > c_{2x}$ ,  $c_{2z_2} < c_{2x}$  cannot occur.

Assume that the above 4 inequalities hold. The first 2 inequalities imply that  $m_{1x}$  has first component  $\geq i+1$ . It must have second component  $\leq j$ , since  $c_{1z_3} = c_{1x}$ . The other 2 inequalities imply that  $m_{2x}$  has first component  $\leq i$ , and second component  $\geq j+1$ , as shown in figure 6.4.

But we are assuming that  $c_{jz_3} = c_{jx}$  for  $j=1,2$ . Thus  $d(y, m_{jx}) < d(z_3, m_{jx})$  for  $j=1,2$ . This is impossible given the locations of  $m_{1x}$  and  $m_{2x}$  in the figure. Thus we have a contradiction, and thus it must be the case that  $z_1$  or  $z_2$  have lower cost than  $x$ .  $\square$

The algorithm to find the root of the SOMT is thus essentially the same as for the case of a single group. First we find the node that is in the center of the smallest grid that contains all the nodes in all the destination lists, and from there if necessary we move to lower cost nodes until we converge to the optimal root.

In the single group case, the root of the tree is always the primary destination of the group. When we have multiple groups, we need to compute the primary

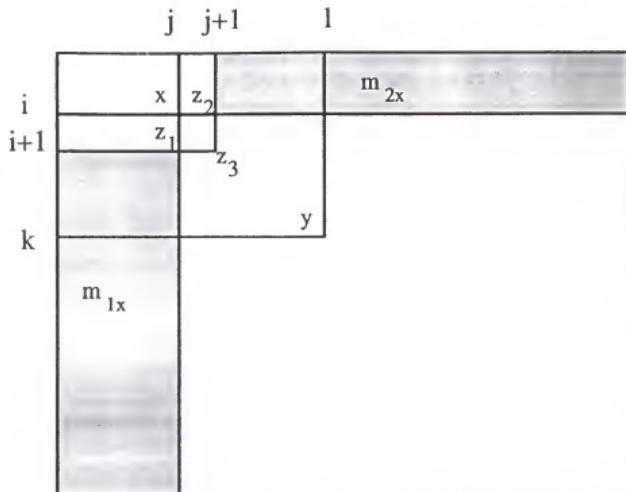


Figure 6.4.

destination for each group. Now a way to compute the primary destinations while the routing is being determined is presented.

### 6.3 Primary Destinations

The following algorithm computes the primary destinations of the  $p$  groups while determining the routing using the heuristic algorithm presented previously.

For a given point  $p$  generated by the algorithm,  $cc_p$  represents the set of multicast groups containing a member of the group that is a descendent of  $p$  in the *SOMT*. For each group  $i$ ,  $NC_i$  denotes the number of disjoint subtrees containing a member group  $i$ . When two points  $p, q$  are joined and substituted by  $\max_o(p, q)$ , every descendent of  $p$  and  $q$  becomes a descendent of  $\max_o(p, q)$ . Let  $g_i$  denote the nodes in the  $i$ th group. Initially,  $cc_p$  includes  $i$  if  $p \in g_i$ . Also, if a group  $i$  does not have any member

that is a descendent of  $p$  or  $q$ , but  $\max_o(p, q) \in g_i$ , then  $i$  must be added to  $cc_{\max_o(p, q)}$ . When all the nodes in group  $i$  belong to the tree, and  $NC_i = 1$ , implying that all of them are descendants of the same point, then the primary destination for this group can be computed. It must be the last node included in the tree, i.e.,  $\max_o(p, q)$ . Let  $D$  now represent the set of distinct nodes in the multicast groups.

Algorithm 6.2 Primary destinations for mesh networks

```

begin
   $cc_p = \emptyset$  for every node  $p$ ;
   $cc_p = cc_p \cup i$  if  $p \in g_i$ ;
  compute optimal root;
  while  $|D| > 1$  do
    let  $p, q$  be substituted by  $\max_o(p, q)$ ;
     $cc_{\max_o(p, q)} = cc_p \cup cc_q$ ;
     $\forall$  group  $i$  not marked do
      if  $\max_o(p, q) \in g_i$ ; then
        mark  $\max_o(p, q)$  in  $g_i$  as visited;
         $cc_{\max_o(p, q)} = cc_{\max_o(p, q)} \cup i$ ;
      if  $NC_i = 0$  and  $\max_o(p, q) \in g_i$  then  $NC_i = 1$ 
      else if  $i \in cc_p$  and  $i \in cc_q$  then  $NC_i = NC_i - 1$ 
      else if  $i \notin cc_p$  and  $i \notin cc_q$  and  $i \in cc_{\max_o(p, q)}$  then  $NC_i = NC_i + 1$ ;
      if  $NC_i = 1$  and all nodes in group  $i$  are marked as visited then
         $PD_i = \max_o(p, q)$ ;
        mark group  $i$ ;
  end;

```

#### 6.4 Multiple Trees

First the subtrees for the connected components of graph  $G'$  as defined previously must be built. These subtrees contains the overlapping nodes, and by maintaining the relative position of the nodes in each subtree the multiple group ordering property is satisfied.

Now we examine how to construct the multicast tree  $T_i$  for group  $i$ . Suppose that the subtrees that must be part of  $T_i$  have been determined. As said before, a depth equal to the maximum distance from the root of each subtree to the farthest node in the subtree that is a member of group  $i$  is assigned to each of these subroots. It may be possible that the same node is root of two or more subtrees. In that case the maximum of the depths is assign to it. The nodes in group  $i$  that did not belong to any of the connected components but that can be inserted in any of the subtrees without increasing the cost of the subtree are included in the subtree. These nodes could increase the traffic generated by the subtree, but not its cost. The rest of the nodes in group  $i$  are assigned a depth of 0. Now these nodes need to be connected together and to the roots of the subtrees. We have a single group with a depth assigned to each node, and an optimal root must be found. A similar algorithm as the one used to find the optimal root for a mesh network when a single group was given can be used. The cost of node  $i$  is now given by  $c_i = \max_{j \in D} \{d(i, j) + \text{depth}(j)\}$ . The same proof discussed in section 6.1 works here. If a node is not an optimal root, then it must have a neighbor with lower cost. Thus the same algorithm used before to converge to an optimal root moving in each iteration to the direction of a lower cost node can be used as well.

Simulations ran for an 8\*8 mesh network under the same conditions as the ones ran for linear and ring networks show similar results. The multiple trees approach shows to be slightly better in terms of the cost of the multicast and much better in

terms of the variability of the primary destinations, while it generates more traffic than the single global tree approach.

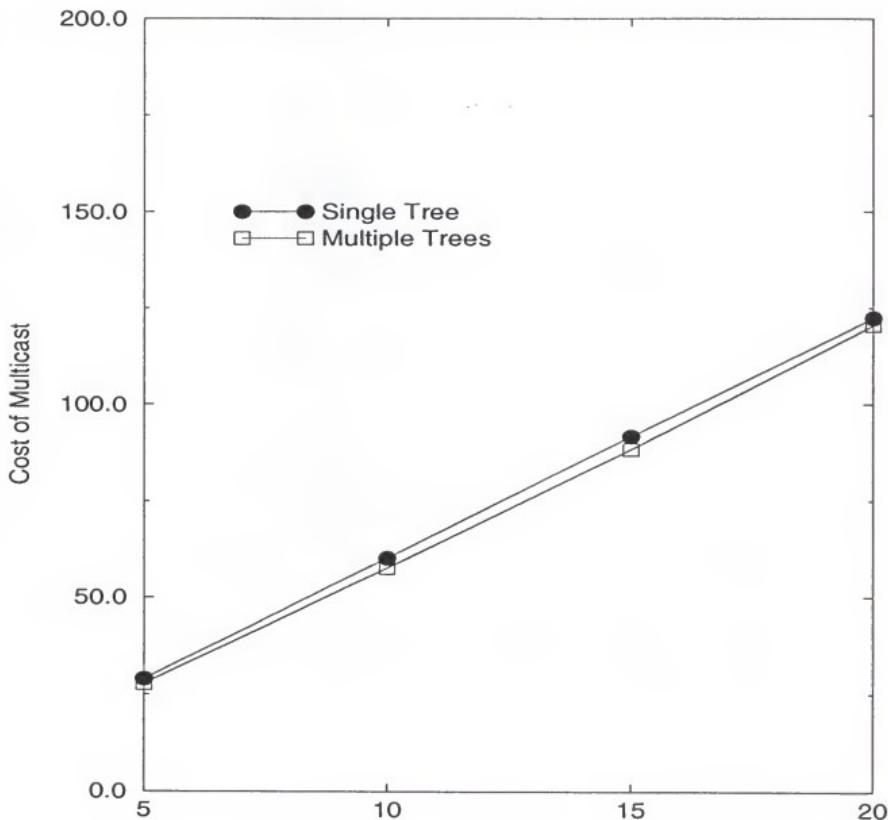


Figure 6.5. Cost for Mesh Network ( $N=64$ )

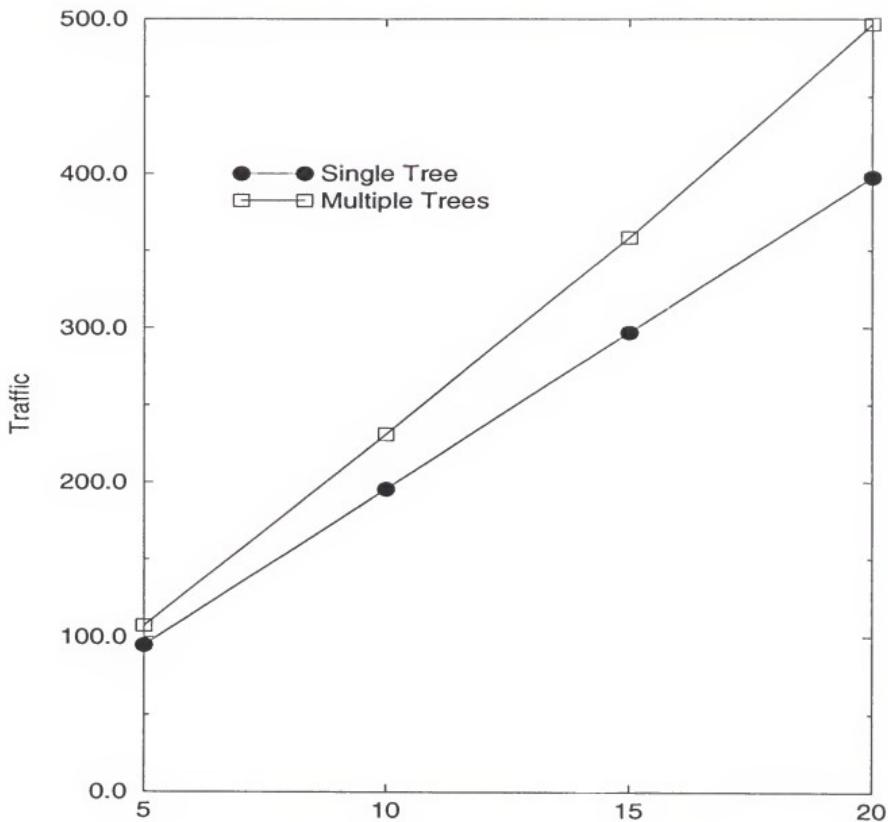


Figure 6.6. Traffic for Mesh Network ( $N=64$ )

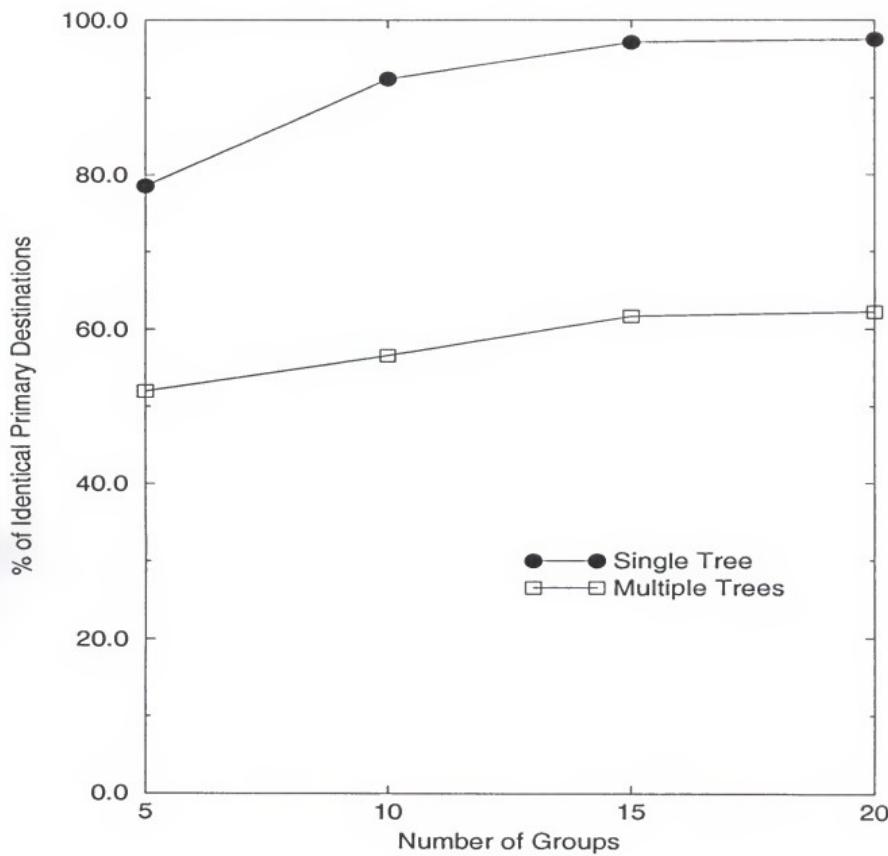


Figure 6.7. Primary Destinations for Mesh Network (N=64)

## CHAPTER 7

### MULTICAST TREES FOR HYPERCUBE NETWORKS

#### 7.1 Single Group

Let  $Q_n$  be a hypercube multiprocessor with  $N = 2^n$  nodes, and let  $D$  be a set of points from the hypercube. A node  $x$  is labeled by the binary number  $b(x) = b_{n-1}b_{n-2}\dots b_0$ . The distance  $d$  in  $Q_n$  between two nodes is equal to the Hamming distance of their binary addresses.

The first step to construct the multicast tree for the set  $D$  is to find the optimal root, that is, the node  $i$  that minimizes  $c_i = \max_{j \in D} d(i, j)$ .

Lemma 7.1 *Let  $Q_s$  be the smallest subcube of  $Q_n$  that contains all the nodes in  $D$ , and let  $x$  be an optimal root. Then  $x \in Q_s$ .*

**Proof:**

Assume, *w.l.g.*, that the first  $n - s$  bits of the nodes in  $D$  are all the same, say all ones. Assume, contrary to the statement of the Lemma, that  $x$  is an optimal root but  $x \notin Q_s$ . Then at least one of the first  $n - s$  bits of  $x$  is 0. Let  $i$  be one of those bit positions. Consider node  $y$  with all bits as  $x$  except bit position  $i$ , in which  $y$  has a 1 instead of a 0. Clearly,  $d(y, a) < d(x, a) \forall a \in D$ . Thus  $c_y < c_x$ , contradicting the fact that  $x$  is optimal. Thus  $x \in Q_s$ , and the Lemma is proved.  $\square$

Let  $D = \{n_1, n_2, \dots, n_m\}$ , where  $m = |D|$ , and let  $Q_s$  be the smallest subcube of  $Q_n$  containing all the nodes in  $D$ . Let  $\bar{n}_i$  denote the *complement* of node  $n_i$ , i.e., the node obtained by reversing every bit in the binary address of  $n_i$  not including the bit positions in which all the nodes in  $D$  agree upon. Note that to form  $\bar{n}_i$  only  $s$  bits are complemented, and that  $d(n_i, \bar{n}_i) = s$ . The cost of the complement of every node in  $D$  is  $s$ . The cost of the neighbors of each complement of every node in  $D$  is at least  $s - 1$ , and so on. This is the idea behind the following algorithm, which computes the optimal root in  $O(N' \log N')$ , where  $N' = 2^s$ .

Algorithm 7.1 Optimal Root

```

begin
  p = {};
  for i=0 to  $N' - 1$  do  $c_i = -1$ ;
  for i=1 to m do
     $c_{\bar{n}_i} = s$ ;
     $p = p \cap \{\bar{n}_i\}$ ;
  while there are nodes with cost -1 do
    q = p;
    p = {};
     $\forall y \in q$  do
       $\forall$  neighbor  $x$  of  $y$  do
        if  $c_x = -1$  then
           $c_x = c_y - 1$ ;
           $p = p \cup \{x\}$ ;
    every node in p is an optimal root
  end;

```

The complexity of this algorithm may be too expensive, especially if there are only a few nodes in the multicast group and  $s = n$ . We propose a heuristic algorithm to find a suboptimal root more efficiently. The main result used to find the optimal root for mesh networks does not always produce the optimal root for hypercubes. A node which is not optimal may have all its neighbors with higher cost than itself.

However, the same idea can be used as a heuristic to find a suboptimal root. The heuristic provides a better result if we start with a good candidate for optimal root.

Let  $n_{ij}$  denote the  $j$ th bit of node  $n_i$ . Define node  $x$  in the following way:  $x_j = 1$  iff  $\sum_{i=1}^m n_{ij} > m/2$ . Node  $x$  has a one in bit position  $j$  if and only if more than half of the nodes in the group have ones in that bit position. This node is called the *median* of  $D$ . In average, we should expect the optimal root to be close to the median. Thus, the median  $x$  of  $D$  is our initial candidate for the root. Then the neighbors of  $x$  are examined to see if any of them has a lower cost than  $x$ . In this case  $x$  is replaced by this neighbor and the process continues. Eventually the algorithm converges to a node that does not have a neighbor with lower cost. The cost of the median can be computed in  $O(mn)$ . Once the cost of a node is computed, the cost of the neighbors can be computed in  $O(m)$ . At most  $n$  hops are required to converge to the final root, so the total complexity of the algorithm is  $O(mn)$ .

We ran simulations for hypercubes going from 2 to 2048 nodes for randomly generated multicast groups. Each node had a fixed probability of 0.2 of being included in the multicast group. The table below shows the results obtained.  $N$  denotes the number of nodes in the hypercube. The second component shows the percentage of cases in which the root computed by the algorithm is in fact the optimal root. The low average in the number of hops required showed in the third column shows that in general, for hypercubes of these sizes, the algorithm converges to the root much faster than the  $O(mn^2)$  worst case time complexity. It is also important to mention that in all the cases (except for one of the runs for  $N = 64$ ) that the heuristic did

not compute the optimal root, the difference between the cost of the optimal and the cost of the node computed by the algorithm was only one. Thus it is fair to say that the heuristic provides a very good approximation for the optimal root.

N	% Optimal Root	Avg # of Hops
2	100	0
4	100	.04
8	100	.11
16	99.2	.22
32	96.9	.42
64	91.9	.53
128	90.5	.62
256	90.9	.61
512	93.8	.79
1024	89.0	.76

Choi and Esfahanian [6] have shown that the *OMT* problem is NP-Complete for hypercubes networks. Lan et al. [28] proposed a heuristic algorithm (*LEN* heuristic) to construct a sub-optimal multicast tree for hypercubes. In their heuristic, the root of the tree is the source of the multicast message.

Each node  $i$  in the tree (beginning with the source) executes a multicast algorithm to select the adjacent nodes to which it will send the message. The multicast group is partitioned into disjoint multicast subgroups, each consisting of destination nodes

which will be descendants of node  $i$  in the tree. The multicast algorithm selects an adjacent node with a maximum number of descendants in the multicast group it received.

Consider the hypercube with  $2^n$  nodes. Assume each node  $v$  is labeled by the binary number  $b(v) = b_{n-1}b_{n-2}\dots b_0$ . The relative address of a node  $b(v_i)$  with respect to a node  $b(v_j)$ , denoted by  $b(v_i|v_j)$ , is defined by taking the bit-wise exclusive-or of the two nodes. Let  $X$  be a collection of  $m$  n-bit binary numbers  $a_{n-1}^k, a_{n-2}^k, \dots, a_0^k$ ,  $1 \leq k \leq m$ , and let  $B(X) = (X_{n-1}, X_{n-2}, \dots, X_0)$ , where  $X_i = \sum_{k=1}^m a_i^k$ . The multicast algorithm executed by node  $v$  upon receiving the multicast subgroup  $X$ , is as follows:

1. For each node  $b(x) \in X$ , replace  $b(x)$  by  $b(x|v)$ .
2. Compute  $B(X) = (X_{n-1}, \dots, X_0)$ , and select the smallest  $i$  such that  $X_i \geq X_j$ , for any  $j$ .
3. If  $X_i = 0$ , stop. Otherwise, create a multicast subgroup  $Y$  containing all  $b(x) \in X$  such that  $b(x)$  has a one in bit position  $i$ .
4. Send  $Y$  to the node in the hypercube adjacent to  $v$  such that  $b(u|v)$  has a one in bit position  $i$ .
5. Set  $X = X - Y$ , and go to step 2.

A problem with this algorithm is that it often splits neighboring nodes among different multicast subgroups, which increases the traffic generated. Choi and Esfahanian have proposed a modification of this algorithm (which they call *Covered heuristic*) that keeps pairs of adjacent nodes together in the same subgroups.

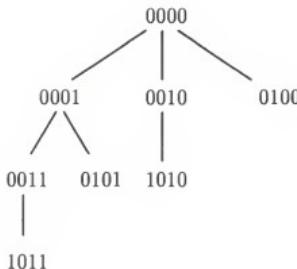
A node  $x$  in a multicast group or destination sublist is *covered* by an adjacent node  $y$  in  $Q_n$  if  $d(x, r) > d(y, r)$ , where  $r$  is the root of the multicast tree being generated. The Covered heuristic is as follows:

1. For each node  $b(x) \in X$ , replace  $b(x)$  by  $b(x|v)$ . Also, mark each node in  $X$  as covered or not covered.
2. Compute  $B(X) = (X_{n-1}, \dots, X_0)$ . Let  $I$  be the set of integers  $i$ , such that  $X_i \geq X_j \forall j = 0, 1, \dots, n-1$ . If  $|I| = 1$  then go to step 3. Otherwise, let  $W = \{b(x) \in X \mid b(x) \text{ is not covered}\}$ . Compute  $B(W)$ , and select the smallest  $i$  such that  $W_i \geq W_j \forall j = 0, 1, \dots, n-1$ .
3. If  $X_i = 0$ , stop. Otherwise, create a multicast subgroup  $Y$  containing all  $b(x) \in X$  such that  $b(x)$  has a one in bit position  $i$ .
4. Send  $Y$  to the node in the hypercube adjacent to  $v$  such that  $b(u|v)$  has a one in bit position  $i$ .
5. Set  $X = X - Y$ , and go to step 2.

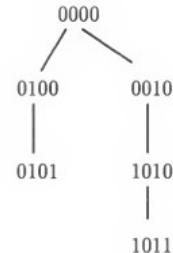
The changes in the second step of the algorithm attempt to keep pairs of adjacent nodes together in the same sublists.

Example 7.1 :

Consider the following multicast group:  $\{0100, 0101, 1010, 1011\}$ . An optimal root for this group is node 0000. The multicast trees generated by each of these heuristics are:

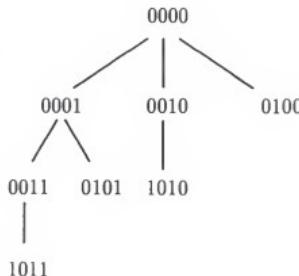


LEN Heuristic

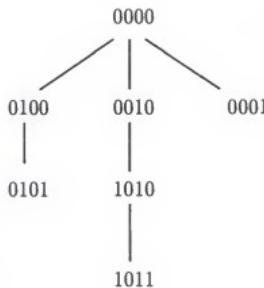


Covered Heuristic

In this case the nodes in  $D$  0101 and 1011 are covered. The Covered heuristic kept these two nodes together with the nodes that covered them, while the LEN heuristic split the pairs of nodes 0100, 0101, and 1010, 1011. However, the Covered heuristic was able to keep these two pairs of nodes together in the same sublists only because  $|I| > 1$  in the second step of the algorithm. It may happen that some nodes could be covered by adjacent nodes in the destination sublist, but they get split because  $|I| = 1$ . Consider, for example, adding the node 0001 to the multicast group in the previous example. Now  $B(X) = \{2, 2, 2, 3\}$  in the first pass of the algorithm, so  $|I| = 1$ , and the following tree is generated:



Nodes 1010 and 1011 are now split, generating a traffic of 7 units. Had we applied the covered idea even when  $|I| = 1$  the following tree would have been generated:



with a traffic of only 6.

This motivates the following modification of the Covered heuristic: in the second step of the algorithm, compute  $B(W)$  and the corresponding value of  $i$  without computing set  $I$ . The new algorithm produces the optimal multicast tree shown in the previous figure for this example.

The performance of this algorithm was compared to the Covered heuristic for randomly generated multicast groups for hypercubes going from 64 nodes to 1024 nodes. The results are shown in the following graph.

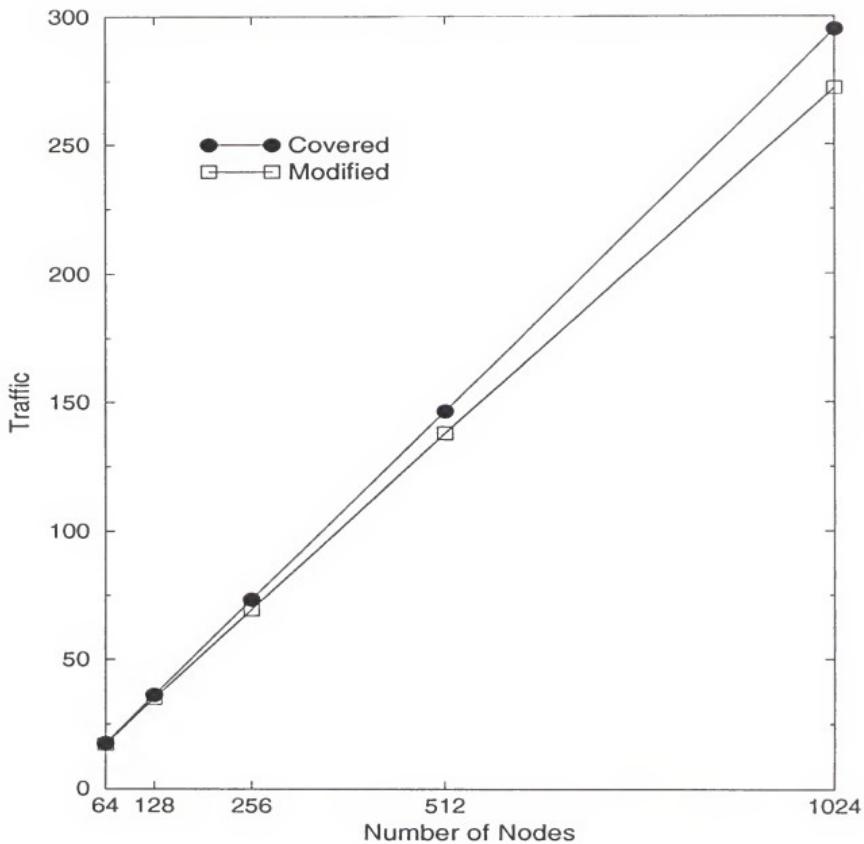


Figure 7.1. Covered heuristic vs Our heuristic

## 7.2 Multiple Groups

The solution to this problem is similar to the solution provided for the single group case. The algorithm to find an optimal root given  $Q_n$  and a multicast group  $D$  can be generalized for  $p$  groups. When a single group was given the cost of each node was computed. The algorithm can be applied for each of the  $p$  groups, and the node  $i$  that minimizes  $c_i = \sum_{j=1}^p \max_{1 \leq k \leq m} d(i, n_{jk})$  can be found. The values given by  $\max_{1 \leq k \leq m} d(i, n_{jk})$  are the values computed by the algorithm for each node, so the node that minimizes the above sum can be found with  $Np$  computations. Thus the optimal root for  $p$  groups has  $O(Np \log N)$  time complexity. A similar algorithm as the heuristic algorithm presented before can also be used for multiple groups to find a suboptimal root.

The multicast tree for the  $p$  groups can be generated using the same heuristic used for a single group which constructs a shortest-path tree and attempts to minimize the traffic generated. The primary destination for each group is computed while the routing is being determined.

## 7.3 Primary Destinations

The following algorithm computes the primary destinations of the  $p$  groups while the routing for the multicast tree is being generated. The heuristic algorithm presented in section 7.1 is used. The primary destination for each group  $i$  is initialized to be the root of the tree. Each time a new sublist is generated by the algorithm, the primary destination for group  $i$  can be set to be the receiver of the sublist (node  $u$  in the algorithm) if and only if the sublist is a subset of  $g_i$ . If the sublist contains

some nodes in  $g_i$  and some that do not belong to  $g_i$ , then  $PD_i$  cannot change any more. Variable  $done_i$  is set to *true*, meaning that this group does not have to be considered any more for the computation of the primary destination. The algorithm is as follows:

1. For each group  $i$ , initialize  $PD_i = root$ ,  $done_i = false$
2. For each node  $b(x) \in X$ , replace  $b(x)$  by  $b(x|v)$ . Also, mark each node in  $X$  as covered or not covered.
3. Let  $W = \{b(x) \in X \mid b(x) \text{ is not covered}\}$ . Compute  $B(W) = (W_{n-1}, W_{n-2}, \dots, W_0)$ , and select the smallest  $i$  such that  $W_i \geq W_j \forall j = 0, 1, \dots, n-1$ .
4. If  $X_i = 0$ , stop. Otherwise, create a multicast subgroup  $Y$  containing all  $b(x) \in X$  such that  $b(x)$  has a one in bit position  $i$ .
5. For each group  $g_i$  with  $done_i = false$  do
  - if  $PD_i = v$  then
    - if  $g_{i1} \in Y$  then
      - if  $g_i \subseteq Y$  then  $PD_i = u$  (as in step 6)
      - else  $done_i = true$
  6. Send  $Y$  to the node  $u$  in the hypercube adjacent to  $v$  such that  $b(u|v)$  has a one in bit position  $i$ .
  7. Set  $X = X - Y$ , and go to step 3.

#### 7.4 Multiple Trees

As before, the subtrees for the connected components of graph  $G'$  as defined previously must be built. Once the subtrees that must be part of the multicast group for

each group  $i$  have been determined, a depth equal to the maximum distance from the root of each subtree to the farthest node in the subtree that is a member of group  $i$  is assigned to each of these subroots. The nodes in group  $i$  that did not belong to any of the connected components but that can be inserted in any of the subtrees without increasing the cost of the subtree are included in the subtree. The rest of the nodes in group  $i$  are assigned a depth of 0. Now these nodes need to be connected together and to the roots of the subtrees. We have a single group with a depth assigned to each node, and an optimal root must be found. The following algorithm computes the optimal root for the multicast tree of group  $i$ :

```

 $c_{opt} = 0; opt\_root = 0;$ 
for i=0 to N-1 do
     $c_i = 0;$ 
    for j=1 to m do
        if  $c_i < d(i, j) + depth(j)$  then  $c_i = d(i, j) + depth(j);$ 
        if  $c_i < c_{opt}$  then
             $c_{opt} = c_i;$ 
             $opt\_root = i;$ 

```

The complexity of this algorithm is  $O(mN)$ , where  $m = \sum_{i=1}^p m_i$  and  $m_i$  is the number of nodes in group  $i$ . A more efficient algorithm can be used to find a suboptimal root using a similar heuristic as the one discussed in the previous section. It can be modified to favor nodes with higher depth. Let  $g_i$  denote the set of nodes in set  $A$  to

be connected and  $depth(i)$  the corresponding depths. The median of the groups is now defined as the node  $x$  where  $x_i = 1$  iff  $\sum_{i=1}^{|A|} g_i * (depth(i) + 1) > \sum_{i=1}^{|A|} (depth(i) + 1)/2$ .

The routing is generated with the same heuristics as discussed in the section 7.1. Simulations ran for  $Q_6$  under the same conditions as the ones ran for the other network topologies discussed in previous chapters show similar results. The multiple trees approach shows to be slightly better in terms of the cost of the multicast and much better in terms of the variability of the primary destinations, while it generates more traffic than the single global tree approach.

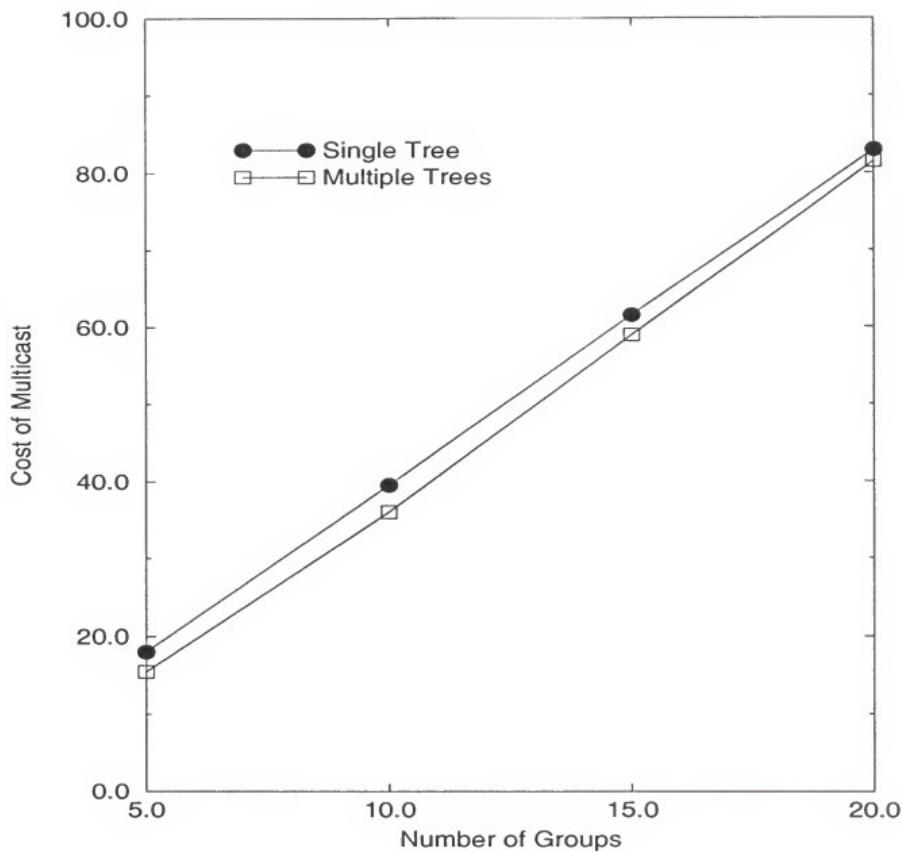


Figure 7.2. Cost for Hypercube Network ( $N=64$ )

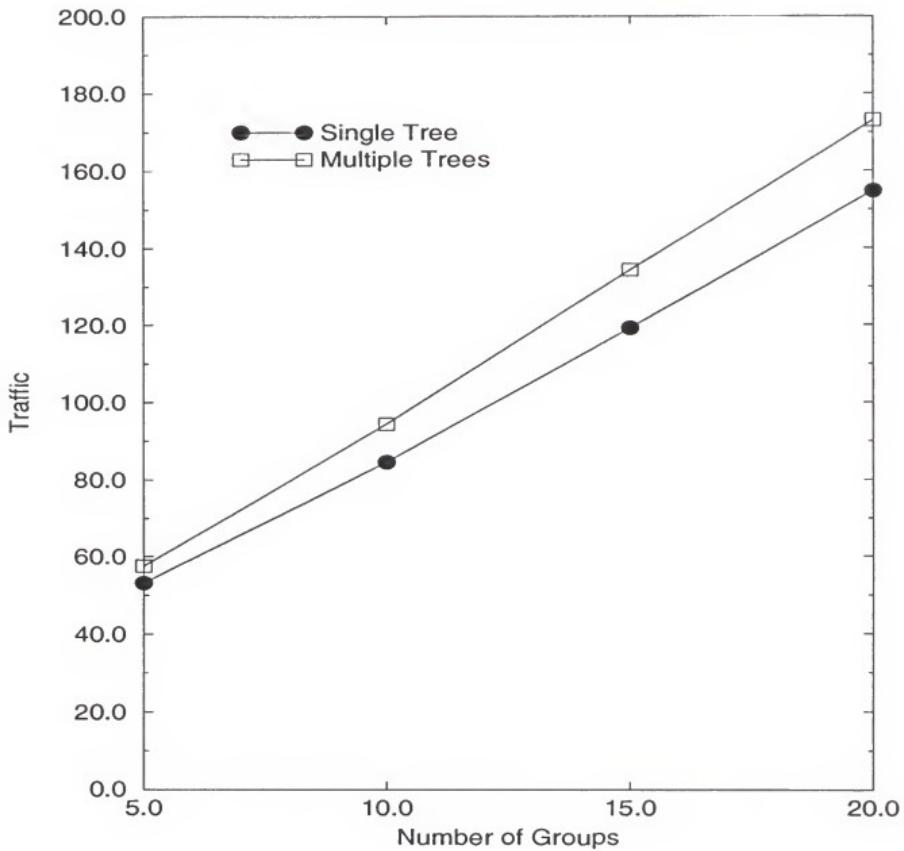


Figure 7.3. Traffic for Hypercube Network ( $N=64$ )

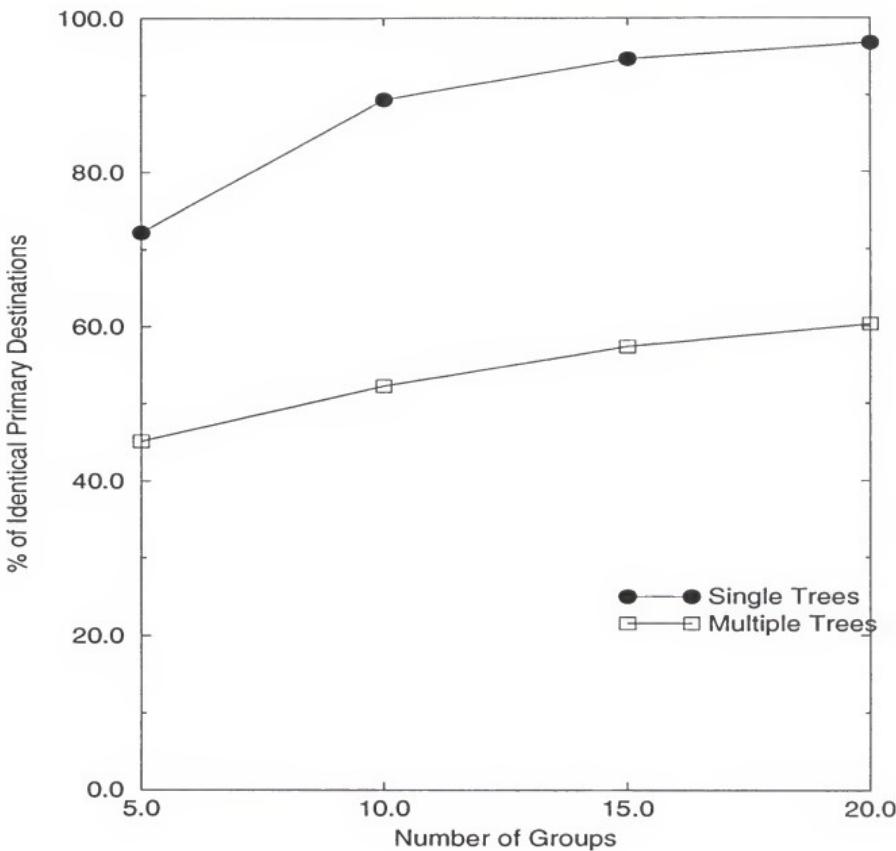


Figure 7.4. Primary Destinations for Hypercube Network ( $N=64$ )

## CHAPTER 8

### CONCLUSION AND FUTURE WORK

The objective of our work was to provide message ordering in a multicast environment given a set of multicast groups. We have proposed multicast trees to achieve this goal. For a single multicast group a single tree is a sufficient and efficient way of providing message ordering. The root of the tree can provide a total order for all the nodes in the group. When multiple groups are given, two graph models can be used to provide the multiple group ordering property: a single global tree can be constructed for all the nodes in overlapping groups, or separate trees can be constructed for each group with common subtrees for overlapping nodes. The goal in both cases is to build trees which minimize time and traffic.

Algorithms to build optimal multicast trees have been discussed for linear and ring networks. The problem of building optimal trees for hypercubes has been shown to be an NP-Complete problem. For mesh networks the problem is conjectured to be also NP-Complete. For these topologies, heuristic algorithms have been proposed that build shortest-path trees while keeping the traffic generated low. For mesh networks, it remains an open problem to determine if there is a polynomial time algorithm to construct an *OMT*. Some similar Steiner tree problems have been shown to be NP-Complete, but their proofs do not seem to be applicable to the *OMT* problem. The

rectilinear Steiner tree problem is also an important problem in determining global routing and wiring for VLSI circuit layout.

For all the network topologies discussed in this work, the comparison made between the single tree approach and the multiple trees approach revealed similar results. The single global tree for all overlapping groups showed a slightly higher cost for the multicasts but generated less traffic than the multiple trees technique. However, the bigger difference, as expected, was in the primary destinations. The multiple trees approach showed much more variability in the primary destinations.

Some related problems were not addressed in this work. For example, the impact that different switching strategies as store-and-forward, virtual cut-through and wormhole may have on the performance of the system when used together with the routing strategies was not considered. Perhaps the major weakness of our approach to provide message ordering is the cost of building the multicast trees. They should be used for multicast groups whose life is relatively long, so the initial cost of building the trees is distributed among many messages. Only static multicast groups were considered in this work. Methods to modify the algorithms to allow for deleting, adding or modifying groups dynamically should be further studied. Another future directions of this work is to consider how to incorporate reliability issues into our scheme. All these problems need to be addressed to develop a complete scheme to be implemented in a real network.

## REFERENCES

- [1] Berry L., "Graph Theoretic Models for Multicast Communications," *Computer Networks and ISDN Systems*, Vol. 20, 1990, pp. 95-99.
- [2] Bharath K., Jaffe J.M., "Routing to Multiple Destinations in Computer Networks," *IEEE Transactions on Communications*, Vol. 31; No. 3, March 1983, pp. 343-351.
- [3] Birman K.P., Joseph T.A., "Reliable Communication in the Presence of Failures," *ACM Transactions on Computer Systems*, Vol. 5, No. 1, February 1987, pp. 47-76.
- [4] Chang J., Maxemchuk N.F., "Reliable Broadcast Protocols," *ACM Transactions on Computer Systems*, Vol. 2, No. 3, August 1984, pp. 251-273.
- [5] Cheriton D.R., Deering S.E., "Host Groups: A Multicast Extension for Datagram Internetworks," *Proceedings of the 9th Data Communications Symposium, ACM SIGCOMM Computer Communications Review*, Vol. 15, No.4, Sept. 1985, pp. 172-179.
- [6] Choi H., Esfahanian A.-H., "A Message-Routing Strategy for Multicomputer Systems," *Networks*, Vol. 22, 1992, pp. 627-646.
- [7] Choi H., Esfahanian A.-H., Houck B., "Optimal Communication Trees with Applications to Hypercube Multicomputers," *Proc. 6th International Conference on Theory and Applications of Graph Theory*, Kalamazoo, MI, 1988.
- [8] Choi H., Esfahanian A.-H., Ni L.M., "One-to-k Communication in Distributed-Memory Multiprocessors," *Proc. 25th Annual Allerton Conference on Communication, Control, and Computing*, Sept. 1987, pp. 268-270.
- [9] Dalal Y.K., Metcalfe R.M., "Reverse Path Forwarding of Broadcast Packets," *Communications of the ACM*, Vol. 21, No. 12, Dec. 1978, pp. 1040-1048.
- [10] Foulds L.R., Graham R.L., "The Steiner Problem in Phylogeny is NP-Complete," *Advances in Applied Mathematics*, Vol. 3, 1982, pp. 43-49.
- [11] Frank A.J., Wittie L.D., Bernstein A.J., "Multicast Communication on Network Computers," *IEEE Software*, Vol. 2, No. 3, 1985, pp. 49-61.

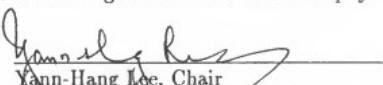
- [12] Garcia-Molina H., Spauster A., "Ordered and Reliable Multicast Communication," Technical Report CS-TR-184-88, Department of Computer Science, Princeton University, October 1988.
- [13] Garcia-Molina H., Spauster A., "Message Ordering in a Multicast Environment," *Proc. 9th Int. Conf. on Distributed Computing Systems*, June 1989, pp. 354-361.
- [14] Garey M.R., Graham R., Johnson D.S., "The Complexity of Computing Steiner Minimal Trees," *Siam J. of Applied Mathematics*, Vol. 32, 1977, pp. 835-859.
- [15] Garey M.R., Johnson D.S., "The Rectilinear Steiner Tree Problem is NP-Complete," *SIAM J. of Applied Mathematics*, Vol. 32, No. 4, June 1977, pp. 826-834.
- [16] Garey M.R., Johnson D.S., *Computers and Intractability*, W.H. Freeman and Company, San Francisco, 1979.
- [17] Garey M.R., Johnson D.S., Stockmeyer L., "Some Simplified NP-Complete Problems," *Theoretical Computer Science*, 1, 1976, pp. 237-267.
- [18] Hakimi S.L., "Steiner's Problem in Graphs and its Implications," *Networks*, Vol. 1, 1971, pp. 113-133.
- [19] Hanan M., "On Steiner's Problem with Rectilinear Distance," *Siam J. of Applied Mathematics*, Vol. 14, 1966, pp. 255-265.
- [20] Ho C.T., Johnson L.S., "Distributed Routing Algorithms for Broadcast and Personalized Communication in Hypercubes," *Proceedings of the 1986 International Conference on Parallel Processing*, August 1986, pp. 640-648.
- [21] Hopcroft J., Tarjan R., "Efficient Planarity Testing," *Journal of the ACM*, Vol. 21, No. 4, October 1974, pp. 549-568.
- [22] Hwang F.K., "The Rectilinear Steiner Problem," *J. Design and Automation Fault Tolerance Anal*, Vol. 2, 1978, pp. 303-310.
- [23] Hwang F.K., Richards D.S., "Steiner Tree Problems," *Networks*, Vol. 22, 1992, pp. 55-89.
- [24] Johnsson L., Ho C.T., "Optimum Broadcasting and Personalized Communication in Hypercubes," *IEEE Transaction on Computers*, Vol. 38, No. 9, September, 1989, pp. 1249-1268.
- [25] Karp R.M., "Reducibility Among Combinatorial Problems," *Complexity of Computer Communications*, R.E. Miller and J.W. Thatcher (eds.), Plenum Press, New York, 1972, pp. 85-104.
- [26] Kaashoek M.F., Tanenbaum A., Hummel S.F., Bal H., "An Efficient Reliable Broadcast Protocol," *Operating Systems Review*, Vol. 23, 1989, pp. 5-19.

- [27] Lamport L., "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM*, Vol. 21, No. 7, July 1978, pp. 558-565.
- [28] Lan Y., Esfahanian A.-H, Ni L., "Multicast in Hypercube Multiprocessors," *Journal of Parallel and Distributed Computing*, Vol. 8, 1990, pp. 30-41.
- [29] Lan Y., Ni L.M., Esfahanian A.-H, "Relay Approach Message Routing in Hypercube Multiprocessors," *Proceedings of the 3rd International Conference on Supercomputers*, May 1988, pp. 174-183.
- [30] Melliar-Smith P.M., Moser L.E., Agrawala V., "Broadcast Protocols for Distributed Systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol 1, No. 1, January 1990, pp. 17-25.
- [31] Rajagopan B., McKinley P., "A Token-Based Protocol for Reliable, Ordered Multicast Communication," *Proc. 9th Int. Conf. on Distributed Computing Systems*, June 1989, pp. 84-93.
- [32] Rao S.K., Sadayappan P., Hwang F., Shor P.W., "The Rectilinear Steiner Arborescence Problem," *Algorithmica*, 7, 1992, pp. 277-288.
- [33] Raynal M., *Distributed Algorithms and Protocols*, John Wiley & Sons, 1988.
- [34] Saad Y., Schultz M., "Topological Properties of Hypercubes," *IEEE Transactions on Computers*, Vol. 37, No. 7, July, 1988, pp. 867-872.
- [35] Scheuermann P., Wu G., "Heuristic Algorithms for Broadcasting in Point-to-Point Computer Networks," *IEEE Transactions on Computers*, Vol. 33, Sept. 1984, pp. 804-812.
- [36] Schneider F., "Synchronization in Distributed Programs," *ACM Transactions on Programming Languages and Systems*, Vol. 4, No. 2, April 1982, pp. 125-148.
- [37] Wall D.W., *Mechanisms for Broadcast and Selective Broadcast*, PhD dissertation, Computer Systems Laboratory, Stanford University, Stanford, California, June 1980.
- [38] Wall D.W., Owicki S., "Construction of Centered Shortest-Path Trees in Networks," *Networks*, Vol. 13, 1983, pp. 207-231.
- [39] Winter P., "Steiner Problem in Networks: A Survey," *Networks*, Vol. 17, 1987, pp. 129-167.

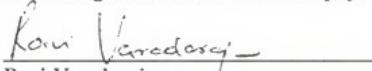
## BIOGRAPHICAL SKETCH

Javier Córdova was born in San Juan, Puerto Rico. He received the B.S. degree in 1981, and the M.S. degree in 1986, both in mathematics, from the University of Puerto Rico at Río Piedras, Puerto Rico. Since 1986 he has been a faculty member in the Computer Science Department at the University of Puerto Rico at Arecibo. Since 1988 he has been in a leave from the University of Puerto Rico to pursue his doctoral studies in computer science at the University of Florida, Gainesville, Florida. After his graduation he plans to resume his duties at the University of Puerto Rico.

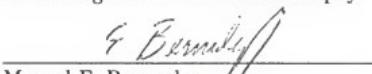
I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

  
\_\_\_\_\_  
Yann-Hang Lee, Chair  
Associate Professor of Computer and  
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

  
\_\_\_\_\_  
Ravi Varadarajan  
Assistant Professor of Computer and  
Information Sciences

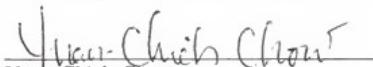
I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

  
\_\_\_\_\_  
Manuel E. Bermudez  
Associate Professor of Computer and  
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

  
\_\_\_\_\_  
Chung-Yee Lee  
Associate Professor of Industrial and  
Systems Engineering

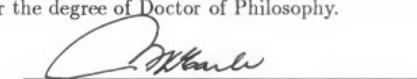
I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Yuah-Chieh Chow  
Professor of Computer and  
Information Sciences

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1993



Winfred M. Phillips  
Dean, College of Engineering

---

Madelyn M. Lockhart  
Dean, Graduate School